UNIVERSIDAD DE PINAR DEL RÍO
"HERMANOS SAÍZ MONTES DE OCA"

"HERRAMIENTAS PARA DISEÑAR Y OPTIMIZAR SISTEMAS DE COSECHA DE MADERA"

Tesis presentada en opción al Título Académico de Master en Nuevas Tecnologías para la Educación

Autor: Ing. Yodeski Rodríguez Álvarez

Tutores: MS c.: Dunieski Pérez Costa[1]
Dr.: Fidel Cándano Acosta[2]
Dr.: Luis Alberto Del Pino[3]

Pinar del Río, 2007

---

1 Master en Informática Aplicada a la Ingeniería y la Arquitectura. Cuba 2000
Master en Geoinformática. Holanda 2003

2 Doctor en Ciencias Forestales. Cuba 2000

3 Doctor en Ciencias de la Física Teórica. México 2002

AGRADECIMIENTOS

Son muchas las personas involucradas en este proyecto. Unas, las que trabajan, merecen el agradecimiento; otras, las que molestan, todo lo contrario.

Quiero agradecer ante todo a mi tutor, amigo y hermano MSc. Dunieski Pérez Costa, por los años de esfuerzo y trabajo, tratando de enseñarme aunque no consiga aprender...; por su paciencia al soportarme.

Quiero agradecer también al resto de mis tutores, al Dr. Fidel Cándano Acosta, a quien le corresponde la idea de este proyecto, por suministrarnos toda la información que necesitamos, por no desesperarse… Al Dr. Luis Alberto Del Pino (Miche), por sus aportes en la matemática.

A mis padres y hermanos, que atendieron mis asuntos cuando redactaba el informe y evitaron ocuparme…

"HERRAMIENTAS PARA DISEÑAR Y OPTIMIZAR SISTEMAS DE COSECHA DE MADERA"

Ing. Yodeski Rodríguez Álvarez

Subdirección Administrativa. Sectorial Municipal de Educación La Palma.
e-mail: yodeski@gmail.com, dperez@af.upr.edu.cu

Resumen

El trabajo realizado consistió en la creación de un conjunto de herramientas (modelo matemático y sistema de aplicación) que permiten diseñar y optimizar sistemas de cosecha de madera evaluando su eficiencia económica. Se aplican como herramientas auxiliares en la asignatura Aprovechamiento Forestal de la carrera de Ingeniería Forestal, de la Universidad de Pinar del Río. Los datos y el formalismo matemático se ajustan a la metodología cubana desarrollada por el Dr. Fidel Cándano Acosta, especialista del propio centro.

Para evaluar la eficiencia de los sistemas de cosecha de madera se usan metodologías que incluyen aspectos como los costos de explotación de las máquinas y herramientas y los rendimientos de las operaciones para las actividades de aprovechamiento de la madera y construcción de caminos y acopiaderos. Estas actividades se relacionan en forma contradictoria por que la disminución de los costos de una eleva los costos de la otra y viceversa.

El modelo matemático permite encontrar los valores para estas actividades que minimizan los costos de los sistemas de cosecha en su totalidad. Es decir, para optimizar los procesos desde el punto de vista de su eficiencia económica.

El software permite diseñar los sistemas de cosecha dando la posibilidad al usuario de definir sus propias fórmulas para el cálculo de los costos de explotación y resolver el modelo matemático. Esto brinda a los estudiantes de la carrera de Ingeniería Forestal la posibilidad de desarrollar habilidades en el uso de medios automatizados en el diseño de sistemas de cosecha de la madera y en la evaluación de su eficiencia económica, respondiendo a uno de los objetivos generales de la asignatura Aprovechamiento Forestal, del tercer año de la mencionada carrera.

Palabras claves: COSECHA/TECNOLOGÍA DE LA MADERA, APROVECHAMIENTO FORESTAL, CAMINOS Y ACOPIADEROS.

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

GLOSARIO DE TÉRMINOS

ActiveX: Tecnología de construcción de componentes de software para proporcionar un estándar de interfase que los haga interoperables para diferentes entornos de programación.

ADO (Access Data Objects): Componente de tecnología ActiveX distribuido por Microsoft Corporation para proporcionar una interfase flexible y eficiente de acceso a orígenes de datos diversos como bases de datos, hojas de cálculos, archivos de correo electrónico, entre otros.

API (Application Program Interfase): Conjunto de elementos programables que permiten gestionar las interfases de los programas de aplicaciones. Ejemplo: API de Windows.

DAO (Data Access Objects): Componente de tecnología ActiveX distribuido como parte del entorno de desarrollo de Microsoft Access para proporcionar una interfase de acceso a bases de datos locales.

Jet Engine: Componente de software que ofrece una interfase para gestionar bases de datos Microsoft Access. Funciona como un driver de dispositivos.

MS-DOS (MicroSoft-Disk Operative System): Sistema operativo creado y distribuido por Microsoft Corporation durante las décadas de 1970 y 1980 para operar arquitecturas IBM-PC, Z-80, entre otras. Tenía una interfase de comandos poco exigente y muy sencilla de utilizar. Era un sistema monotarea.

ODBC (Open DataBase Connection): Protocolo de acceso a orígenes de datos desarrollado y distribuido por Microsoft Corporation en forma de componente de software para hacer interoperables el mayor número posible de estos. ODBC ofrece una interfase única de acceso a orígenes de datos tales como Microsoft SQL Server, Paradox, Oracle, entre otros.

OLE DB (Objects Linked and Embeded for DataBases): Protocolo de acceso a orígenes de datos desarrollado por Microsoft para sustituir ODBC y distribuido en forma de componente de software para lograr la interoperabilidad de orígenes de datos diversos como bases de datos de diferentes compañías como Microsoft, Paradox y Oracle; así como documentos creados en diferentes tipos de aplicaciones como hojas de cálculo, textos, correos electrónicos, formato ASCII, entre otros.

RDO (Remote Data Objects): Componente de tecnología ActiveX distribuido como parte del entorno de desarrollo de Microsoft Access para proporcionar una interfase de acceso a bases de datos remotas, o sea en aplicaciones cliente servidor.

SQL (Structured Query Language): Es un lenguaje para construir consultas a bases de datos relacionales que ha devenido el estándar internacional usado por todos los SGBD relacionales. Está avalado por la Organización Mundial de Estándares (ISO), y es proporcionado no solo como lenguaje de desarrollo "puro" sino también como metalenguaje incorporado a modelos de acceso a datos distribuidos por diferentes compañías.

UML (Unified Modelling Language): Lenguaje de modelado unificado. Es un medio para modelar comprende un conjunto de reglas semánticas aplicables al análisis y diseño de bases de datos, software, flujos productivos, entre otros.

VBA (Visual Basic for Applications): Componente de software distribuido por Microsoft Corporation como alternativa de lenguaje de para programación para entornos de desarrollo de grandes aplicaciones como la suite Office, AutoCAD, ArcGIS, entre otros.

1. INTRODUCCIÓN

El aprovechamiento forestal de un país está condicionado por factores de tipo socioeconómico, ergonómico, geomorfológico y de masa, entre otros. Sobre la base de estos factores se crean metodologías para calcular los costos unitarios de las tecnologías de aprovechamiento forestal. Estas tecnologías están formadas por una secuencia de operaciones que se realizan en el bosque (tala, desrame, troceado, carga y transporte) y la secuencia de operaciones para la construcción de caminos y acopiaderos. Así, los costos de la tecnología dependen de los costos de la secuencia de operaciones en el bosque y de los costos de construcción de caminos y acopiaderos.

Los costos de las tecnologías determinan su eficiencia, los niveles de rentabilidad y las ganancias derivadas del aprovechamiento de madera; por ello, este aspecto es abordado en varias investigaciones realizadas por FAO [7], Sessions et-al. [18], Stokes et-al. [20], Greene et-al. [10], Silva et-al. [19] y Lanford et-al. [15].

La evaluación de la eficiencia de los sistemas de cosecha, sus costos y su optimización tiene que analizarse de forma integral y no tratar de minimizar el costo parcialmente porque puede conducir a errores. Digamos, el desrame de los árboles en el área de tala puede encarecer esta operación, sin embargo, aumenta la carga útil del tractor arrastrador, su rendimiento y por tanto, reduce los costos de la extracción. Otro ejemplo se aprecia en la construcción de caminos; mejorar los caminos aumenta el costo de su construcción, pero simultáneamente los camiones de transporte aumentan las velocidades y su rendimiento, disminuyendo entonces los costos del transporte. Estos aspectos, se manifiestan de forma opuesta o contradictoria, y el problema que generan solo se resuelve usando la modelación matemática como herramienta.

En la clase de aprovechamiento Forestal que aborda estas cuestiones, se requiere del uso de la modelación matemática y de sistemas que automaticen la resolución de estos modelos. La resolución manual requeriría de mucho tiempo, partiendo de que primeramente se deben crear los sistemas de cosecha, y luego, resolver los problemas de contradicción existentes entre los aspectos del aprovechamiento antes mencionados, lo cual trae implícita la posibilidad de cometer errores en los cálculos.

En correspondencia con lo expresado anteriormente, el **problema** de esta investigación radica en cómo lograr un conjunto de herramientas que permitan diseñar sistemas de cosecha de madera y evaluar su eficiencia, encontrando variantes óptimas de aprovechamiento forestal con costos mínimos, desde las clases de Aprovechamiento Forestal de la Carrera de Ingeniería Forestal de la Universidad de Pinar del Río.

A tales efectos, se considera que el **objeto** de la investigación se enmarca en *la Ejecución del proceso de creación y optimización de sistemas de cosecha de madera en las clases de Aprovechamiento Forestal, de la carrera de Ingeniería Forestal.*

El **objetivo general** es entonces, viabilizar el proceso de creación de sistemas de cosecha y optimización de sus costos en la asignatura de aprovechamiento forestal de la carrera de Ingeniería Forestal, derivando de este los siguientes **objetivos específicos**:

1. Crear un modelo matemático que permita optimizar los costos de los sistemas de cosecha de madera.
2. Crear un sistema computacional que permita:
   - ✓ diseñar sistemas de cosecha de madera.
   - ✓ calcular los costos de las tecnologías de aprovechamiento forestal.
   - ✓ resolver el modelo matemático de forma automatizada.
   - ✓ suministrar herramientas de edición, visualización y de intercambio de información sobre sistemas de cosecha.

Contribuyendo todo al entrenamiento de futuros ingenieros en la toma de decisiones en la actividad de explotación de bosques.

Derivado del análisis y la relación entre el problema, objeto y objetivos de la investigación se establecen las siguientes **Ideas a defender**, que guían la presente investigación:

1. La modelación matemática, como herramienta para la optimización de procesos económicos y de la producción resuelve los problemas de contradicción que existen entre los elementos que se utilizan para evaluar la eficiencia de los sistemas de cosecha de madera permitiendo encontrar variantes tecnológicas óptimas.
2. La resolución de modelos matemáticos bajo condiciones productivas solo es posible y viable usando herramientas computacionales que garanticen obtener soluciones rápidas y confiables.
3. El uso de herramientas que permitan diseñar y modelar procesos productivos y de toma de decisiones puede resultar útil en la formación y capacitación de recursos humanos.

Para dar cumplimiento a los objetivos se desarrollaron las siguientes tareas:

1ra. Etapa: Fundamentación del problema.

1. Diagnóstico de la situación actual sobre el uso de la modelación matemática y las herramientas computacionales para resolver problemas de optimización en la actividad de Aprovechamiento forestal.
    a. Análisis y estudio de las metodologías de aprovechamiento que se usan en Cuba.
    b. Análisis de los modelos matemáticos y herramientas computacionales usados para resolver problemas de optimización de variantes tecnológicas.
    c. Observación de actividades del proceso.
2. Caracterización de la evolución histórica de los modelos matemáticos y herramientas computacionales en el contexto Internacional y en Cuba.

2da. Etapa: Fundamentación teórica del modelo matemático y de clases propuestos para la solución del problema de optimización y el diseño de la aplicación, así como de las tecnologías de software utilizadas.

1. Análisis y estudio de métodos para resolver modelos matemáticos.

2. Análisis de las técnicas de programación orientada a objetos.
    a. Determinación de los tipos abstractos a usar y propuesta de un modelo de clases para la implementación de la solución computacional.

3era. Etapa: Creación del modelo matemático y diseño e implementación de la solución computacional.

1. Crear un modelo matemático para optimizar tecnologías aprovechamiento forestal.
2. Diseño e implementación de un sistema automatizado para diseñar sistemas de cosecha de madera, resolver el modelo matemático y calcular los costos de las tecnologías de aprovechamiento forestal.

El desarrollo de las tareas de investigación fue posible mediante el empleo de los siguientes métodos:

El enfoque dialéctico – materialista integral permitió:

- Establecer el carácter desarrollador y contradictorio de las relaciones entre los componentes de las tecnologías de aprovechamiento forestal.
- Analizar las contradicciones y componentes en el objeto y en el campo de investigación (proceso de toma de decisiones).
- Definir el modelo que permite resolver la contradicción.
- Determinar como se da la relación causa – efecto dialécticamente.
- Descubrir nuevas cualidades del modelo y herramienta propuestos.
- Integrar métodos teóricos y empíricos.

Entre los métodos teóricos se utilizaron:

- El histórico-lógico para la determinación de particularidades, tendencias, regularidades del proceso de toma de decisiones en la actividad de aprovechamiento forestal.
- El sistémico estructural para fundamentar el modelo a partir de la determinación de los componentes que lo conforman, sus relaciones, estructura y diseño general.

- La modelación posibilitó la construcción del modelo matemático y de clases para la optimización y solución computacional respectivamente.

Como procedimientos de los métodos teóricos se utilizaron el *análisis - síntesis* y la *inducción - deducción* en la interpretación de la información documental para la determinación de antecedentes, así como la obtención de las tendencias que caracterizan el comportamiento del proceso de creación de sistemas de cosecha, y en general tributaron a la elaboración de la fundamentación teórica de la investigación.

Entre los métodos empíricos utilizados se encuentran:

- Entrevista a profesores y especialistas en aprovechamiento forestal y estudiantes de la especialidad Ingeniería forestal, cuyo objetivo fue fundamentar el problema.
- Observación a clases para caracterizar la organización espacio – temporal del proceso de creación y evaluación de eficiencia de sistemas de cosecha de madera.

El **aporte teórico** fundamental de la investigación es el modelo matemático que permite evaluar sistemas de cosecha de madera y encontrar variantes tecnológicas de aprovechamiento forestal óptimas sobre la base de la minimización de los costos de las tecnologías.

Como **aporte práctico** se obtiene un sistema que automatiza el diseño y creación de sistemas de cosecha y la resolución del modelo, permitiendo calcular los costos mínimos de las tecnologías.

2. CARACTERIZACIÓN DEL PROBLEMA

2.1 Identificación y caracterización del problema

Crear un sistema de cosecha de madera requiere del análisis de factores que involucran los costos de las tecnologías de aprovechamiento forestal, las cuales a su vez dependen de los costos de las secuencias de operaciones que se realizan para cosechar la madera y de los equipos que se utilizan. Para el cálculo de estos costos existen metodologías que difieren de un país a otro, incluso de una región a otra dentro del mismo país, condicionadas por factores de suelo, geomorfológicos, geográficos, entre otros. Las formas de cálculo varían entre un sistema y otro. Por tal motivo, la evaluación de los sistemas de cosecha y cálculo manual de los costos de las tecnologías es prácticamente imposible, teniendo en cuenta que entre los aspectos que determinan estos costos existen contradicciones que solo pueden ser resueltas usando modelos matemáticos, los cuales a su vez requieren de herramientas computacionales para su resolución.

En las clases de aprovechamiento forestal del tercer año de la carrera de Ingeniería forestal, se estudian los aspectos conceptuales y prácticos del proceso de aprovechamiento de la madera. Este proceso se concibe como una secuencia de operaciones en las cuales intervienen factores humanos y tecnológicos. Para planificar el proceso es necesario disponer de información sobre existencias de madera (proveniente del ordenamiento forestal), disposición geográfica de los sitios a explotar, de las industrias de procesamiento, así como los parámetros técnicos de los equipos y herramientas y de la propia industria.

El estudiante recibe los aspectos conceptuales necesarios para llevar a cabo esta planificación teniendo en cuenta todos estos elementos. Esta planificación se convierte en un Sistema de Cosecha del cual es necesario calcular los parámetros económicos necesarios para establecer su nivel de eficiencia. El objetivo final sería diseñar el sistema de forma tal que este nivel de eficiencia sea óptimo.

El diseño del sistema de cosecha tiene que tener en cuenta las operaciones que pueden ser incluidas en la secuencia del proceso. Estas operaciones son de diferente naturaleza en función de las actividades de aprovechamiento y de construcción de caminos y acopiaderos. Las operaciones de las etapas se realizan con herramientas y

maquinarias. Tanto las etapas en sí como estas herramientas y maquinarias tienen un grupo numeroso de parámetros técnicos que se utilizan para calcular rendimientos y costos de explotación respectivamente.

Entre las actividades de aprovechamiento y construcción de caminos y acopiaderos se busca un compromiso entre las distancias medias de los caminos y acopiaderos que se van a construir para obtener el costo óptimo de la tecnología.

Estos problemas requieren la solución de un formalismo matemático bastante complicado. La solución en busca del óptimo implica la ejecución de gran cantidad de cálculos que incluye el empleo de métodos numéricos. Este volumen de procesamiento es prácticamente imposible de realizar manualmente.

No es objetivo de la asignatura profundizar en estos formalismos o que el estudiante aprenda a realizarlos, sino proveerlo de herramientas que le permitan concentrarse en la planificación del proceso e interpretar los resultados.

2.2 Solución del problema con el empleo de las TIC

La optimización de los sistemas de cosecha solo es posible mediante el empleo de un modelo matemático como herramienta. Éste tiene que ser capaz de resolver el compromiso entre las distancias medias entre caminos y acopiaderos para encontrar los valores de éstas que hacen mínima la suma de los costos de las dos actividades, y por lo tanto, óptimo el sistema de cosecha.

El diseño de sistemas de cosecha, el cálculo de sus parámetros económicos y la resolución del modelo, solo es posible mediante el empleo de una herramienta computacional. Ésta tiene que ser capaz de gestionar de manera organizada un gran volumen de información sobre los equipos y herramientas disponibles, las operaciones asociadas a las etapas así como los sistemas de cosecha que se van creando y que quedan establecidos como tecnología.

Para solucionar el problema que nos ocupa creamos un modelo matemático usando métodos numéricos y diseñamos y creamos un software que permitan a los estudiantes del tercer año de la carrera Ingeniería Forestal, diseñar visualmente y evaluar la eficiencia de sistemas de cosecha de madera, parametrizando las secuencias de operaciones y los equipos usados para acometer estas.

El software es capaz de gestionar todos los datos; calcular los costos unitarios de explotación de los equipos y los rendimientos de las operaciones asociadas a las etapas, con lo que se determina el costo de la tecnología; y encontrar los valores de distancias medias entre caminos y acopiaderos que la optimizan.

2.3 Modelo conceptual del problema

El modelo conceptual está constituido por un grupo de componentes interrelacionados. Estos vínculos son de diferente naturaleza y tiene que ver con los roles de cada uno de los componentes dentro del objeto.

Los componentes definidos dentro del modelo conceptual y sus particularidades se describen a continuación:

Clase de aprovechamiento: Espacio de intercambio entre el profesor y los estudiantes en el que se manifiesta el problema.

Estudiantes: Entidad receptora de la solución del problema.

Profesor: Entidad ejecutora de la solución.

APROV PLUS: Herramienta computacional. Es el medio para solucionar el problema.

Diseñar Sistemas de cosecha: Habilidad que desarrolla el estudiante en la clase a través del medio computacional.

Optimizar Sistemas de cosecha: Habilidad que desarrolla el estudiante en la clase a través del medio computacional.

Aprovechamiento: Componente de la habilidad *Diseñar Sistemas de cosecha*.

Caminos y acopiaderos: Componente de la habilidad *Diseñar Sistemas de cosecha*.

Modelo matemático: Método para resolver *Compromiso entre S y L.*

Entre las entidades del modelo conceptual se establecen vínculos cuya descripción aparece a continuación:

Participa: | *Clase de aprovechamiento forestal* & *Estudiantes*

El estudiante es uno de los elementos del espacio de interacción de la solución del problema que constituye la clase de aprovechamiento.

Participa: | *Clase de aprovechamiento forestal* & *Profesor*

El profesor es el otro elemento del espacio de interacción de la solución del problema que constituye la clase de aprovechamiento,

Usa: | *Estudiantes* & *APROV PLUS*

El estudiante usa APROV PLUS para diseñar y optimizar sistemas de cosecha.

Supervisa: | *Profesor* & *APROV PLUS*

El profesor supervisa el trabajo del estudiante con el medio para diseñar y optimizar sistemas de cosecha de madera.

Permite: | *APROV PLUS* & *Diseñar Sistemas de cosecha*

APROV PLUS permite diseñar de manera visual e interactiva sistemas de cosecha de madera.

Permite: | *APROV PLUS* & *Optimizar Sistemas de cosecha*

APROV PLUS permite optimizar sistemas de cosecha de madera.

Contiene: | *Diseñar Sistemas de cosecha* & *Aprovechamiento*

Las operaciones de aprovechamiento son componentes dentro del sistema de cosecha de madera.

Contiene: | *Diseñar Sistemas de cosecha* & *Caminos y acopiaderos*

Las operaciones de construcción de caminos y acopiaderos son componentes dentro del sistema de cosecha de madera.

Compromiso entre S y L: | *Aprovechamiento* & *Caminos y acopiaderos*

Al disminuir la distancia media entre caminos (S), y la distancia media entre acopiaderos (L), se reduce el costo de aprovechamiento y aumenta el costo de construcción de caminos y acopiaderos. Al aumentar la distancia media entre caminos y la distancia media entre acopiaderos ocurre lo contrario. Como el costo de la tecnología es la suma de los costos de las dos actividades, S y L expresan la relación entre ellas y es necesario buscar los valores que producen el mínimo costo a nivel de tecnología.

Resuelve: | *Modelo matemático & Compromiso entre S y L*

El modelo matemático resuelve el compromiso entre las distancias medias entre caminos y acopiaderos.

Las interacciones entre los distintos elementos del modelo conceptual se reflejan en la Figura 2.1

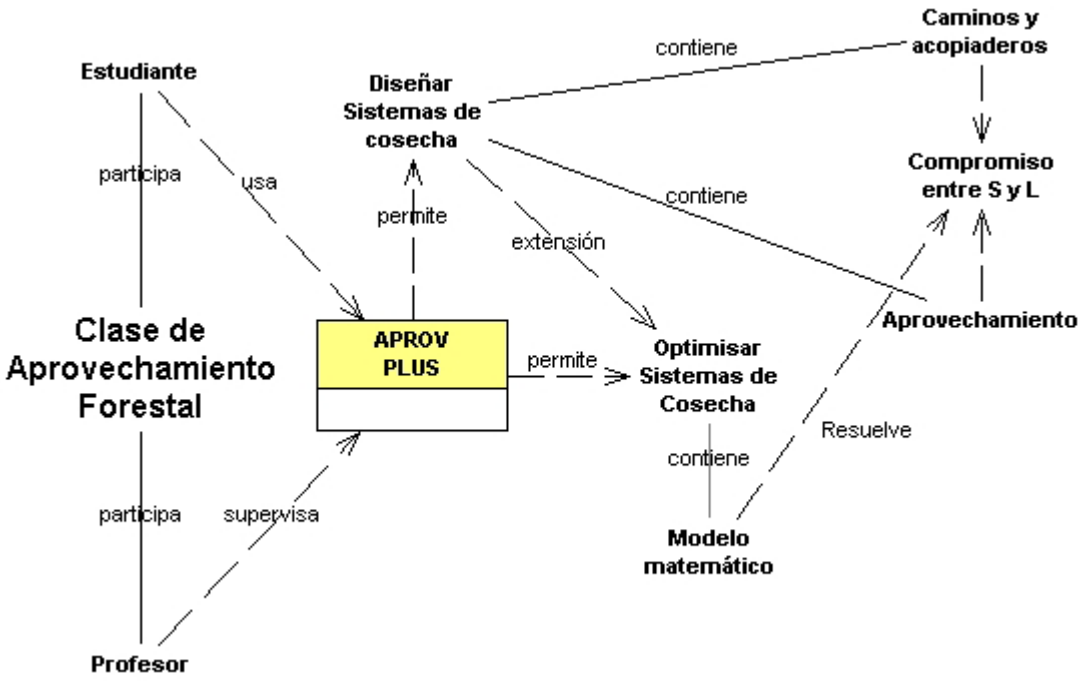


Figura 2. 1 Diagrama de interacción de elementos del modelo conceptual

2.4 Análisis de viabilidad y costo de la propuesta

Antes de llevar a cabo la tarea de diseño del software APROV PLUS, se realizó una valoración aproximada de su costo y tiempo de desarrollo con uso del Modelo de Diseño Temprano de COCOMO II (Constructive Cost Model). Se comparó el costo con los beneficios que se obtendrían con el empleo del Sistema Automatizado para la Evaluación de Sistemas de Cosecha de Madera y se determinó acometer la tarea de diseñarlo e implementarlo.

Para la estimación del costo se calcularon los indicadores siguientes con uso del software USC COCOMO II del Centro para Ingeniería del software de la Universidad de California:

Entradas Externas (EI): entrada de usuario que proporciona al software diferentes datos orientados a la aplicación (Tabla 2.1).

Tabla 2. 1. Entradas Externas

Nombre	**Cantidad de ficheros**	**Cantidad de Elementos de datos**	**Complejidad**
Equipos	4	1	Medio
Parámetros de costos fijos	1	7	Bajo
Parámetros de costos de operación	1	10	Bajo
Parámetros de costos de labor	1	7	Bajo
Datos para optimizar costos	0	8	Bajo

Salidas Externas (EO): salida que proporciona al usuario información orientada de la aplicación. En este contexto la "salida" se refiere a informes, pantallas, mensajes de error, etc. (Tabla 2.2).

Tabla 2. 2. Salidas Externas

Nombre	**Cantidad de ficheros**	**Cantidad de Elementos de datos**	**Complejidad**
Resumen de costos unitarios	0	4	Bajo
Gráfico de la secuencia activa	0	4	Bajo
Gráfico de la tecnología	0	4	Bajo
Cuadro Resumen de la secuencia activa	1	7	Bajo
Cuadro Resumen de la tecnología	1	7	Bajo
Diagrama del sistema de cosecha	0	15	Bajo

Peticiones (EQ): son entradas interactivas que resultan de la generación de algún tipo de respuesta en forma de salida interactiva (Tabla 2.3).

Tabla 2. 3. Peticiones

Nombre	Cantidad de ficheros	Cantidad de Elementos de datos	Complejidad
Operaciones	2	15	Medio
Asignación de equipo	1	8	Bajo

Ficheros internos (ILF): son archivos (tablas) maestros lógicos (o sea una agrupación lógica de datos que puede ser una parte de una gran base de datos o un archivo independiente) (Tabla 2.4).

Tabla 2. 4 Ficheros Internos

Nombre	Cantidad de registros	Cantidad de Elementos de datos	Complejidad
Equipos	50+	3	Medio
Tractor	5	28	Medio
Herramienta	5	28	Medio
Camión	5	28	Medio
Animal	2	28	Medio
Etapa	15	8	Medio
Secuencia	2	3	Bajo
Tecnología	20	7	Medio
Parámetros del rendimiento	10	5	Medio

Según los datos anteriores se registraron los puntos de función que se muestran en la Figura 2.2

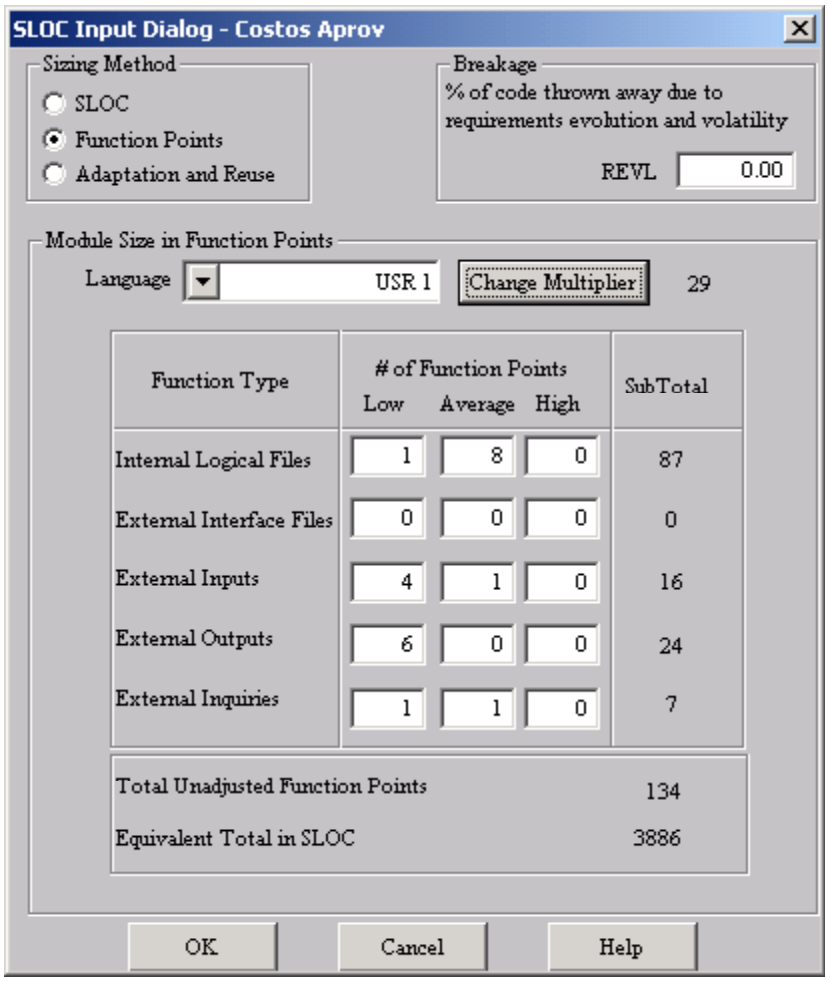


Figura 2. 2. Líneas de código empleadas.

Se consideró como entorno de programación Delphi 7, tomándose como promedio 29 líneas código por punto de función (según tabla de reconciliación de métricas consultada), obteniéndose así 3886 instrucciones fuentes con un Total de Puntos de Función Desajustados de 134.

La Tabla 2.5 muestra los valores considerados de los Multiplicadores de esfuerzo (EM) para el Modelo de Diseño Temprano.

Tabla 2. 5. Valores de los EM

Factores	Valor	Justificación
RCPX	0.83 (Bajo)	Base de Datos simple.
RUSE	0.95 (Bajo)	El nivel de reuzabilidad es a través del programa.
PDIF	0.87 (Bajo)	El tiempo y la memoria estimada para el proyecto son de baja complejidad.
PREX	1 (Normal)	Los especialistas tienen experiencia en el uso de las tecnologías
CIL	1 (Normal)	Se han utilizado herramientas de alto nivel de desarrollo como el Delphi 7y ModelMaker.
SCED	1 (Normal)	Los requerimientos de cumplimiento de cronograma son normales.
PERS	1 (Normal)	La experiencia del personal de desarrollo es normal, tienen una buena capacidad.

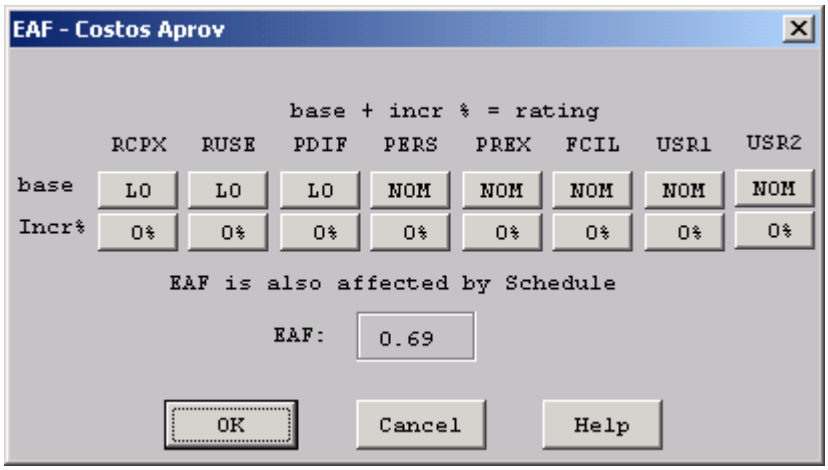


Figura 2. 3. Valores de Multiplicadores de Esfuerzo.

Los valores considerados de los Factores de escala (SF) se muestran en la Tabla 2.6.

Tabla 2. 6. Valores de los SF

Factores	Valor	Justificación
PREC	2.48 (Alta)	Se posee una buena comprensión de los objetivos del producto, se tiene experiencia en la realización de software de este tipo.
FLEX	3.04 (Normal)	Debe haber considerable cumplimiento de los requerimientos del sistema.
TEAM	1.10 (Muy alto)	El equipo que va desarrollar el software es cooperativo.
RESL	4.24 (Normal)	Existe un plan definido.
PMAT	4.68 (Normal)	Se encuentra en el nivel 2 (normal).

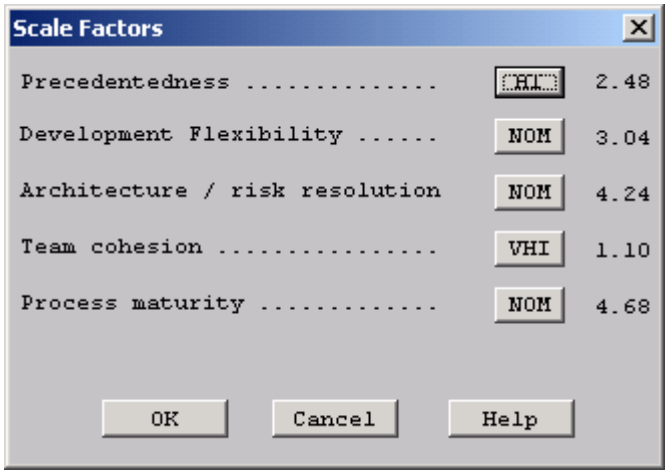


Figura 2. 4. Factores de Escala.

Considerándose un salario promedio de $350 se obtuvieron los siguientes resultados que aparecen en la Figura 2.5.

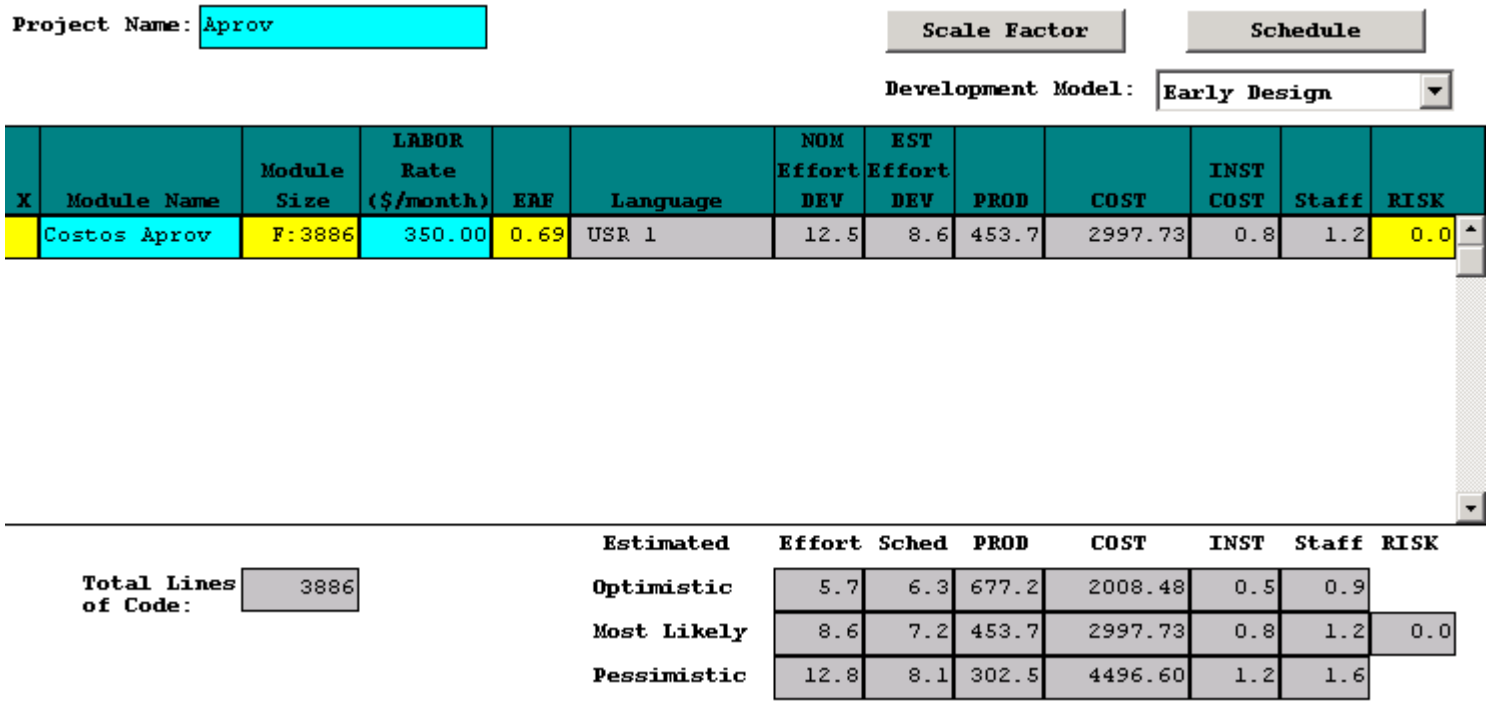


Figura 2. 5. Resultados. Ventana de Cálculos de Cocomo II.

De donde se obtiene:

Esfuerzo (DM).

$$DM = \frac{(ValorOpimista + 4(ValorEsperado) + ValorPesimista)}{6}$$

$$DM = \frac{(5.7 + 4(8.6) + 12.8)}{6}$$

DM = 8.82 Hombres / Mes

Tiempo (TDev):

$$TDev = \frac{(ValorOptimista + 4(ValorEsperado) + ValorPesimista)}{6}$$

$$TDev = \frac{(6.3 + 4(7.2) + 8.1)}{6}$$

TDev = 7.2 Meses

Cantidad de hombres (CH):

$$CH = \frac{DM}{TDev}$$

$$CH = \frac{8.2}{7.2}$$

CH = 1.23 hombres

Costo de la Fuerza de Trabajo:

$$CTP = \frac{(ValorOptimista + 4(ValorEsperado) + ValorPesimista)}{6}$$

$$CTP = \frac{(2008.48 + 4(2997.73) + 4496.60)}{6}$$

CTP = $ 3082.66

Cálculo de costo de los medios técnicos: costo de utilización de los medios técnicos.

CMT= Cdep + CE + CMTO

Donde:

Cdep: Costo por depreciación (se consideró 0).

CMTO: Costo de mantenimiento de equipo (se consideró 0 porque no se realizó).

CE: Costo por concepto de energía.

CE = HTM x CEN x CKW

Donde:

HTM: Horas de tiempo de máquina necesarias para el proyecto.

CEN: Consumo total de energía

CKW: Costo por Kw. / horas ($0.09 hasta 100 Kw., $ 0.30 de 101 a 150 Kw., $ 0.40 de 151 a 200 Kw., $ 0.60 de 201 a 250 Kw., $ 0.80 de 251 a 300 Kw. y 1.30 más de 300 Kw.)

HTM= (Tdd x Kdd + Tip x Kip) x 152

Donde:

Tdd: Tiempo promedio utilizado para el diseño y desarrollo (7 meses).

Kdd: Coeficiente que indica el promedio de tiempo de diseño y desarrollo que se utilizó en la máquina (0.50)

Tip: Tiempo utilizado para las pruebas de implementación (4 horas).

Kip: Coeficiente que indica el % de tiempo de implementación utilizado en la máquina. (0.8)

HTM = (7x0.50+4 x 0.8) x152

HTM = (3.50+ 3.2) x 152

HTM = 1018.4 H

CEN = 0.608 Kw. / h (Estimado)

KW = HTM x CEN

KW = 1018.4 x 0,608

KW = 619.18

CKW = (100 x 0.09)+ (50 x 0.30)+ (50 x 0.40) + (50 x 0.60)+ (50 x 0.80) + (319 x 1.30)

CE = $528.70

Por lo antes considerado el costo de los medios técnicos es:

CMT = $528.70

Cálculo del Costo de Materiales:

En el cálculo de los costos de los materiales se consideró el 5 % de los costos de los medios técnicos.

CMAT = 0.05 x CMT

Donde:

CMT: Costo de los medios técnicos.

CMAT = 0.05 x $528.70

CMAT = $26.44

Después de realizados los cálculos correspondientes a los Costos Directos (CD), se obtienen los siguientes resultados:

CD= CPT+CMT+CMAT

CD= 3082.66+528.70+26.44

CD= $3637.80

Costo Total del Proyecto: Para calcular el valor total del proyecto se utilizó la siguiente expresión:

CTP = CD + 0.1 x SB

CTP = 3637.80 + 0.1 x 3082.66

CTP = $3946.07

El software que se propone está dirigido al proceso de diseño y optimización evaluación de la eficiencia de los sistemas de cosecha de madera y obtener las variantes tecnológicas óptimas con un costo mínimo, sobre la base de parámetros establecidos acorde con las metodologías cubanas. Este se utilizará en principio en la docencia aunque es aplicable en la producción (dependiendo de la voluntad organizativa de estas entidades), por tanto sus beneficios son de orden social y económico. Su utilización en la docencia permitirá modelar y evaluar la eficiencia de sistemas de cosecha de madera, permitiendo su optimización y dando como resultado los costos mínimos de las tecnologías de aprovechamiento forestal.

Recursos Humanos:

Dos personas para el análisis, diseño y desarrollo del sistema:

Tutor: MSc. Dunieski Pérez Costa.

Autor: Yodeski Rodríguez Álvarez.

Recursos Técnicos:

Hardware para su diseño y desarrollo:

Procesador: Pentium II 500 Mhz.

Memoria: 192 MB

Disco Duro: 60 Ghz

Unidad de Respaldo: CD- RW

Monitor: Resolución SVGA (800 x 600) píxeles.

Software:

Sistema Operativo Windows 98 o Superior.

Microsoft Access 2000.

Borland Delphi 7.

Model Maker.

3. CREACIÓN, DISEÑO E IMPLEMENTACIÓN DE LAS SOLUCIONES

3.1 Valoración y justificación de la solución matemática

El problema de optimización de la tecnología de aprovechamiento forestal surge a partir de la contradicción entre los costos unitarios de aprovechamiento (*Cu*), y los costos unitarios de la construcción de caminos y acopiaderos (*Cusl*).

El primero se calcula sumando los costos unitarios de las etapas de aprovechamiento:

$$Cu = \sum_{i=1}^{n} \left(\frac{C \exp}{\mathrm{Re}\, nd} \right)$$

Donde:

Cexp	: Costo de explotación de la máquina utilizada ($/hora)
Rend	: Rendimiento de la máquina en la etapa (m^3/hora)
n	: Número de etapas de la secuencia de aprovechamiento

El segundo se calcula por la fórmula:

$$Cusl = \frac{(10 * Cr * L) + (Cl * 10^4)}{V * S * L}$$

Donde:

Cr	: Costo de construcción de caminos ($/km)
Cl	: Costo de construcción de un acopiadero ($)
V	: Volumen de madera a aprovechar (m^3/ha)
S	: Distancia media entre caminos (m)
L	: Distancia media entre acopiaderos (m)

La secuencia de aprovechamiento siempre incluye la etapa de extracción. En esta etapa el rendimiento (*Rend*), se calcula utilizando la fórmula siguiente:

$$\mathrm{Re}\, nd = \frac{V * (60 - Ti)}{\frac{da}{Vrsc} + Ta + \frac{da}{Vrcc} + Td}$$

Donde:

V	: Volumen promedio de la carga de la máquina por ciclo (m^3)
Ti	: Tiempo de interrupción del trabajo de la máquina (min./hora)
da	: Distancia promedio de extracción (m)
Vrsc	: Velocidad media de recorrido sin carga (m/min)
Ta	: Tiempo de amarre de la madera (min)
Vrcc	: Velocidad media de recorrido con carga (m/min)
Td	: Tiempo de desamarre de la madera (min)

Para calcular *da* se utiliza la expresión:

$$da = \frac{K}{3} * \left(\sqrt{\left((F * S)^2 + L^2 \right)} + \sqrt{\left(\left(0.5 * F * S^2 \right) + \left(0.5 * L^2 \right) \right)} \right)$$

Donde:

K	: Coeficiente de sinuosidad (curvatura de las vías de extracción)
F	: Valor relacionado con el sentido de la extracción
S	: Distancia media entre caminos (m)
L	: Distancia media entre acopiaderos (m)

La etapa de extracción es la única en la secuencia de aprovechamiento que depende de *S* y *L*. El costo unitario de construcción de caminos y acopiaderos también depende de estos parámetros. Sin embargo esta dependencia se manifiesta en proporcionalidad inversa.

Cuando se construyen más caminos y acopiaderos disminuye la distancia promedio de extracción y por tanto el costo unitario de la etapa y de la secuencia de aprovechamiento completa. Pero entonces aumenta el costo unitario de la construcción de caminos y acopiaderos. Dado que el costo de la tecnología es la suma de estos dos costos, el aumento o disminución de *S* y *L* produce efectos contrarios en ellos.

Por esta razón es necesario utilizar las herramientas de programación matemática para encontrar los valores de *S* y *L* que proporcionan el costo unitario óptimo de la tecnología, en este caso mínimo.

Al sumar las expresiones de los costos de la secuencia de aprovechamiento y de construcción de caminos y acopiaderos, se obtiene una función donde los parámetros ajustables (variables independientes) son *S* y *L* (los demás permanecen constantes), de la forma:

$$Ct(S,L) = Cu(S,L) + Cusl(S,L)$$

Donde:

Ct(S,L)	: Costo de la tecnología en función de las distancias *S* y *L*
Cu(S,L)	: Costo unitario de aprovechamiento en función de las distancias *S* y *L*
Cusl(S,L)	: Costo unitario de la construcción de caminos y acopiaderos en función de las distancias *S* y *L*

El problema se reduce a encontrar el mínimo absoluto de la función. Si se dispusiera de las formas triviales de las ecuaciones para calcular los rendimientos de las etapas en las secuencias de aprovechamiento y de construcción de caminos y acopiaderos, se hubiera podido utilizar un método tan simple como el de los multiplicadores de Lagrange. Entonces se hubiera podido descomponer la función de dos variables en una de una variable y una restricción, derivar la expresión resultante e incluir solo las ecuaciones obtenidas para ser evaluadas.

Sin embargo, el programa da al usuario la posibilidad de construir sus propias fórmulas para calcular los rendimientos y elimina la posibilidad de contar con tales ecuaciones de antemano. Entonces hay que trabajar con fórmulas que se definen en tiempo de ejecución, cuyas derivadas

no se pueden calcular a menos que se disponga de una herramienta lógica para hacerlo, tarea sumamente complicada, por no decir imposible…

Estas fórmulas son resueltas por un analizador algebraico desarrollado con este propósito. El analizador fue validado para expresiones complejas comparando sus resultados con los del que proporciona MS Excel.

La alternativa escogida es la de un método numérico. Consiste en evaluar la función en el intervalo bidimensional que se define para los rangos de *S* y *L* establecidos por el usuario. Hay que aclarar que en la realidad del diseño de tecnologías de aprovechamiento las distancias *S* y *L* varían normalmente en un rango de valores discretos entre 200 y 2000 metros, lo que da un tiempo de computación más que aceptable en la corrida del modelo.

En las pruebas realizadas en PC's con parámetros estándar (entre 1 y 3 GHz de frecuencia del CPU, y entre 64 y 256 MB de memoria RAM), el tiempo de corrida estuvo alrededor de los cinco segundos.

Ideas básicas del modelo propuesto

La función *Ct(S,L)* se trata como una serie numérica de la forma:

$$Min\left(\sum_{S=a}^{b}\sum_{L=c}^{d}\left(Cu(S,L)+Cusl(S,L)\right)\right)$$

Donde:

$$S = a + n * \Delta S,...,b$$

$$n = 0,...,\frac{b-a}{\Delta S}$$

$$L = c + n * \Delta L,...,d$$

$$n = 0,...,\frac{d-c}{\Delta L}$$

a y *b* son los extremos del intervalo que acota a *S*, *d* y *c* acotan a *L*. ΔS es el incremento asignado para evaluar la función en el intervalo de *S*, ΔL es el incremento para evaluar el intervalo definido para *L*.

Si como en este caso no está disponible de antemano la expresión de la función, o simplemente es una serie numérica que no puede ser normalizada, la búsqueda del extremo mínimo se puede resolver siguiendo los pasos siguientes:

1. Asignar a los incrementos en el dominio (ΔS y ΔL), un valor razonablemente pequeño de acuerdo a los propósitos del problema.
2. Acotar las variables independientes (*S* y *L*) en intervalos.

3. Evaluar la función de forma iterativa en el intervalo definido utilizando el incremento asignado.
4. Localizar el menor valor obtenido al evaluar la función.

Este valor es el mínimo de la función en el intervalo definido y es el óptimo, donde el costo unitario de la tecnología es menor que en cualquier otro punto de la región. Los valores de *S* y *L* donde se encuentra el mínimo son las distancias medias que optimizan el costo de la tecnología.

Solución operacional del modelo

El tiempo de computación para la corrida del modelo depende en este caso del valor asignado al incremento. El sistema calcula el incremento para independientemente de las longitudes de intervalo definidas para S y L, el número de iteraciones sea el mismo.

De manera que al analizar la región tridimensional descrita por el conjunto de vectores (*S*, *L*, *Ct(S,L)*), realiza dos operaciones.

1. Evalúa los puntos de *S* y *L* a la distancias de sus respectivos incrementos, cubriendo la región con un retículo (Fig. 3.1) donde cada nodo corresponde al vector (*Si*, *Lj*, *f(Si,Lj)*). Al terminar localiza el nodo con el mínimo valor.
2. Realiza el mismo análisis sobre el entorno de puntos alrededor del nodo localizado hasta los nodos contiguos en las tres direcciones y los dos sentidos, pero evaluando las variables independientes valor por valor. Al terminar localiza el punto con el menor valor de *f(S,L)*.

Este procedimiento no garantiza al cien por ciento que el mínimo sea encontrado. Puede suceder que la función tenga un comportamiento asintótico en algún punto del retículo que logre "engañar" el rastreo. Sin embargo al analizar las expresiones comúnmente usadas se ha encontrados que son continuas en los intervalos de *S* y *L* que razonablemente se pudieran definir. Una expresión construida por el usuario que tuviera comportamientos extraños en estos intervalos, sencillamente es más probable que carezca de sentido.

Se puede implementar fácilmente un procedimiento que rastree la totalidad de los valores de los intervalos en busca del mínimo *Ct(S,L)* absoluto, pero esto equivaldría a multiplicar el tiempo de computación a cambio de abarcar variantes casi seguramente incorrectas.

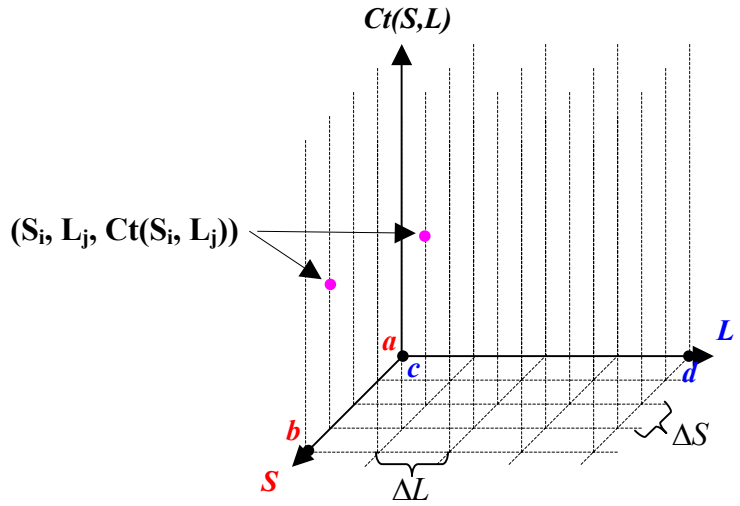


Figura 3. 1 Evaluación de los puntos de S y L

3.2 Valoración crítica de sistemas afines

Actualmente en la asignatura de aprovechamiento forestal se utilizan dos software para optimizar tecnologías de sistemas de cosecha: el sistema PACE y el sistema SECTAM.

El sistema PACE (The Production and Cost Evaluation Program) es un producto creado en Canadá que tiene 21 años de existencia. Fue creado para funcionar sobre el sistema operativo *MS-DOS*, que era el estándar más extendido entonces para IMB-PC. Se utilizó Borland Turbo Pascal versión 3.0 en su construcción.

PACE permite construir un sistema de cosecha definiendo un número fijo de etapas para la actividad de aprovechamiento de la madera: tala, extracción, carga y transporte. Tiene un módulo para introducir parámetros de máquinas y herramientas que se usan en cada de una de las etapas. Diseñar la tecnología consiste en primer término en asignar a cada etapa la herramienta o la máquina que se prevé utilizar. Si el sistema de la tecnología incluye la construcción de caminos y acopiaderos entonces el sistema ofrece una secuencia de operaciones también fija para que el usuario seleccione la herramienta o máquina que se usará en cada operación. Un sistema de cosecha o tecnología se completa combinando estas dos actividades.

Los parámetros introducidos para cada máquina o herramienta permiten calcular sus costos unitarios de explotación. Para las etapas de ambas actividades el usuario tiene

que introducir valores a un grupo de parámetros para calcular el rendimiento de cada una. El sistema calcula los costos por metro cúbico de madera para cada etapa y para la tecnología y muestra todos los resultados en varios cuadros resúmenes, divididos por actividades (aprovechamiento y caminos / acopiaderos). Para cada tecnología se puede optimizar la distancia media entre caminos y acopiaderos.

Los datos sobre máquinas / herramientas y actividades se pueden editar para modificarlos, eliminarlos o introducir nuevos. Los datos se almacenan en archivos en disco, un archivo para cada máquina o herramienta, un archivo para cada actividad y de forma similar para cada tecnología.

La interfase del sistema es eficiente y cómoda desde el punto de vista de los estándares del sistema operativo *MS-DOS* (Figura 3.1). Estos estándares, sin embargo, no están a la altura de los sistemas de interfase gráfica como Windows, lo cual es reconocido e innecesario de argumentar.

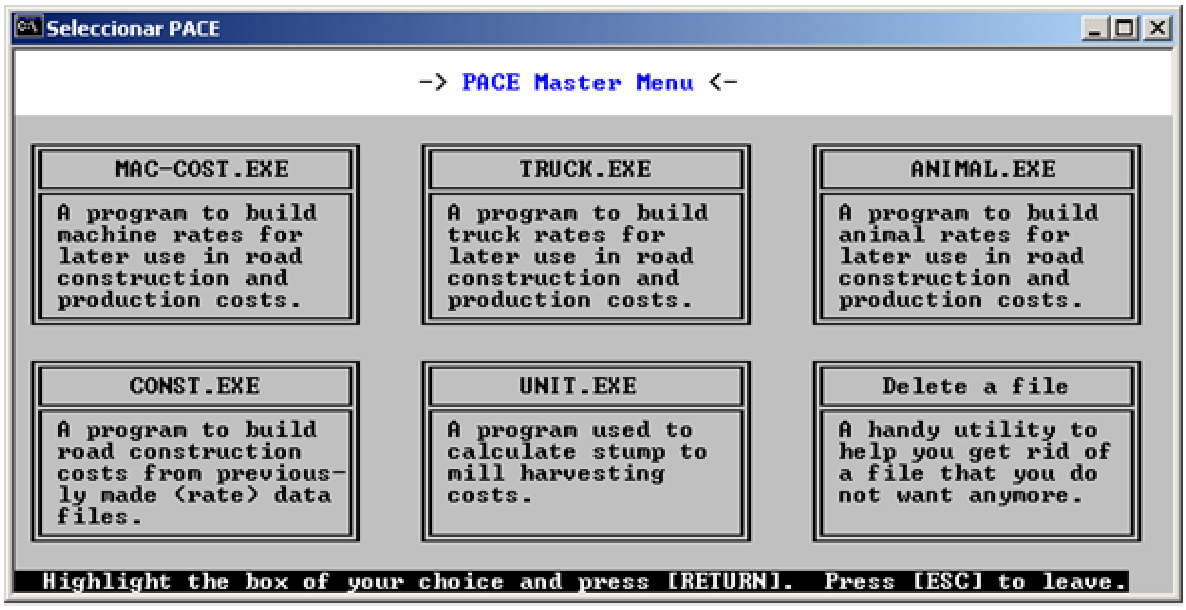


Figura 3. 2. Ventana principal del Sistema PACE

La base de datos está ordenada en forma de archivos dispersos, lejos del concepto moderno de contenedor de objetos de base de datos en archivo único, predominante actualmente para aplicaciones de oficina. Para su acceso, actualización y control de la integridad de los datos no se usa lenguaje relacional o de otro enfoque de manipulación

de datos sino rutinas de tratamiento de archivos de propósito general. Hemos comprobado que es muy fácil inducir errores de integridad al manejar los archivos erróneamente. Otro ejemplo de falla en la integridad es la imposibilidad de restringir las máquinas / herramientas para ciertos tipos de operaciones asociadas a etapas. Por ejemplo, no hay forma de evitar que una motosierra sea asignada como máquina / herramienta a la etapa de transporte.

Aparte de esto, la mayor limitante del sistema para su aplicación a nuestras condiciones es que opera con un formalismo matemático que es resultado de los estudios realizados sobre los bosques de algunas regiones de Canadá, donde las condiciones de explotación de los equipos y el rendimiento de las operaciones no se ajustan siempre a nuestras condiciones. En este sentido es necesario aclarar que si se pueden parametrizar las máquinas / herramientas y el rendimiento de las etapas pero no se pueden modificar las fórmulas matemáticas.

Por último, las facilidades de edición se ven limitadas por que al asignar los equipos a las operaciones no se pueden modificar dentro del archivo de actividades creados. Ante cualquier cambio necesario hay que crear una nueva actividad, introducir toda la información desde el principio y almacenar en un nuevo archivo.

De manera general se puede decir que PACE es útil para desarrollar nociones de automatización en el diseño y optimización de sistemas de cosechas pero no garantiza adaptabilidad a nuestras condiciones económicas y tecnológicas, además de algunos problemas de seguridad e integridad de los datos.

El sistema SECTAM (Sistema de Explotación y Costos de Tecnologías de Aprovechamiento de la Madera), fue creado con el mismo propósito de diseñar y optimizar sistemas de cosecha. También funciona sobre el sistema operativo *MS-DOS* empleando sus estándares óptimos, incluyendo la funcionalidad del mouse, ausente en el caso de PACE. Su descripción operacional es en esencia la misma que la realizada para el PACE.

Desde el punto de vista de los datos aquí fue utilizado un sistema de gestión de base de datos para diseñarlos, el Nantucket Clipper Summer'85. El programa también fue creado

usando su entorno de desarrollo. Para manipular los datos se usa lenguaje *SQL*, que lo cual mejora considerablemente el control de su integridad.

El formalismo matemático de SECTAM es resultado de estudios realizados [4], en áreas forestales de la provincia de Pinar del Río teniendo en cuanta las normas tecnológicas de explotación y las características organizativas de la actividad de cosecha en cuba y las condiciones naturales de nuestros bosques. En este sentido se adapta mucho mejor a las necesidades de formación de los estudiantes.

Sus limitaciones principales, haciendo a un lado la imposibilidad de aprovechar las ventajas de las interfases gráficas de Windows, son dos. En primer lugar la falta de una funcionalidad para agregar o eliminar etapas de las actividades. Esto es hoy una posibilidad que dan las máquinas multipropósito. En segundo lugar la falta de otra funcionalidad para cambiar la forma trivial de las expresiones algebraicas que calculan cada uno de los indicadores de rendimiento de las etapas. Esto es hoy una necesidad debido al aumento de la complejidad de las actividades por la introducción de estas máquinas y de variantes organizativas diferentes.

En general SECTAM es un sistema más apropiado para nuestras condiciones pero carece de la flexibilidad necesaria para modelar las complejidades que surgen en el diseño de los sistemas de cosecha.

3.3 Justificación de la elección del tipo de Software creado

El sistema APROV PLUS intenta adaptarse al diseño de sistemas de cosechas en medio de condiciones tecnológicas y organizativas cambiantes para desarrollar en el estudiante no solo la habilidad para usar software en el diseño y optimización de cosechas, sino a cambiar creativamente el diseño en función de una adaptabilidad a los cambios en el objetivo de lograr la eficiencia económica.

Para lograrlo APROV PLUS incorpora un grupo de características que se pueden resumir en los puntos siguientes, y que contienen novedades y mejoras respecto a los sistemas anteriormente descritos.

- ✓ Uso del concepto de contenedor de base de datos en un archivo único, lo más apropiado en aplicaciones de oficina.
- ✓ Diseño de datos relacional optimizado.
- ✓ Modelo avanzado en la gestión de los datos.
- ✓ Posibilidad de establecer flexiblemente la secuencia de etapas en los dos tipos de actividades.
- ✓ Posibilidad de construir las expresiones algebraicas para calcular el rendimiento de cada etapa.
- ✓ Capacidad de visualización y exportación de los datos en tablas y gráficos.

Los datos que gestiona el programa están almacenados en una base de datos Microsoft, en un archivo de extensión MDB con seguridad a nivel de usuario. El sistema ofrece funciones para crear y restaurar copias de la base de datos. Debido a la necesaria parametrización para calcular costos de equipos y rendimientos de las etapas, que requirió la definición de varios sistemas de clasificación, se consideró conveniente no exportar los datos de las tecnologías en archivos a menos que en futuros desarrollos se haga necesario. Esto requeriría funcionalidades adicionales para unificar codificadores.

El diseño de los datos fue sometido a un proceso de normalización para comprobar la calidad de los datos. Las relaciones entre las entidades son permanentes para garantizar la integridad referencial por medio del controlador *Jet Engine*. Los otros tipos de integridad están garantizados por los propios elementos de diseño y por propiedades de las tablas, *Jet Engine* mediante.

Para gestionar los datos el programa utiliza el modelo de acceso a datos *ADO*. Las operaciones sobre los datos se realizan en buena medida sobre *SQL* incorporado al modelo como metalenguaje. De manera que se eliminan problemas relacionados con la integridad que van más allá de lo que puede controlar el *Jet Engine* como la asignación correcta de los tipos de máquinas / herramientas a las operaciones de las etapas o las reglas de cálculo de los costos de explotación en función de que una máquina / herramienta sea alquilada o no. En resumen, han sido incluidas como parte de los datos un grupo de reglas de inferencia para evitar al usuario errores de interpretación y ambigüedades durante la parametrización de equipos y operaciones.

El programa consta de una lista de etapas que abarcan las labores tradicionales más las combinaciones de labores posibles realizadas por máquinas multipropósito disponibles en la actualidad. Todos los flujos posibles de operaciones de etapas que tienen carácter lógico se pueden establecer. Al elegir una etapa el estudiante tiene siempre ante sí la lista de etapas posible para continuar el flujo de la cosecha de forma lógica. Por ejemplo, es posible que después de talar pueda extraer o desramar entre otras posibilidades, y dependiendo de lo que escoja, dispondrá de nuevo de la operación de desrame después de la extracción o incluso de la transportación. Las etapas se pueden eliminar en un punto de la secuencia, con lo que se corta esta, después de lo cual el sistema sigue ofreciendo la lista de etapas según los flujos lógicos posibles. Estas relaciones se modelaron definiendo grafos para modelar las relaciones entre las etapas en las actividades y matrices de adyacencia para resolver los grafos (controlar los flujos).

APROV PLUS tiene en su base de datos las fórmulas que de acuerdo a los estudios realizados [4], mejor calculan el rendimiento de cada operación asociada a cada actividad. Sin embargo, incorpora una herramienta para que el usuario construya sus propias fórmulas y actualice los valores de los parámetros incluidos dentro de ellas. Entonces el usuario tiene la alternativa de utilizar las fórmulas por defecto o incluir las suyas propias. Al guardar la tecnología se incorporan las nuevas fórmulas para futuras sesiones de trabajo. El procesamiento de las fórmulas se realiza con un analizador sintáctico algebraico similar al de MS Excel, que fue validado tomando a este como patrón.

Los dos puntos anteriores desde el punto de vista conceptual y de la solución al problema de investigación constituyen las mejoras más importantes y por lo tanto los aportes de mayor valor respecto a los sistemas PACE y SECTAM.

Las posibilidades de visualización y exportación de los datos comprenden las tablas resúmenes y los gráficos. Además, estos se pueden exportar a libros MS Excel para realizar con ellos cualquier tipo de análisis deseado.

Las tablas se construyen por actividades y por tecnologías. En ambos casos se ordenan los costos de explotación de los equipos en sus tres categorías y generales, y los

rendimientos de las etapas en las columnas. Por las filas se ordenan cada una de las etapas para el caso de las actividades de aprovechamiento y caminos / acopiaderos por separado, y para el caso de la tecnología las etapas de aprovechamiento más una fila resumen de las etapas de caminos / acopiaderos. Los gráficos se construyen en estas mismas variantes.

3.4 Tecnologías empleadas

Una componente importante y fundamental dentro de las TIC lo constituyen las tecnologías del software, como soporte lógico para el procesamiento y gestión de la información fundamentalmente.

En el proceso de diseño y creación de productos informáticos están involucrados un conjunto no muy conservador de sistemas que van desde complejas herramientas para modelar el proceso de creación del software, hasta sistemas de tratamiento gráfico para lograr interfases cómodas y agradables garantizando además una imagen corporativa capaz de atraer a posibles usuarios interesados.

En la creación del software propuesto utilizamos para modelar el proceso de desarrollo un sistema de terceros incorporado a la versión 7 de Borland Delphi conocido como ModelMaker, el cual cuenta con un conjunto de 10 tipos de diagramas basados en el estándar *UML*, permitiendo una integración completa con Delphi para la creación de clases y Units.

Como soporte de bases de datos utilizamos Microsoft Access 2002, teniendo en cuenta que es un Sistema de Gestión de Bases de datos relacional y su novedosa característica de contenedor de datos en fichero único. La modelación de los datos se realizó usando el enfoque relacional, específicamente el Modelo Entidad Relación.

Como lenguaje de programación se escogió la implementación de Borland Inprice del Object Pascal en el entorno de desarrollo Borland Delphi cuyas características se exponen más adelante. Para acometer el acceso a los datos utilizando este lenguaje se usó el Modelo de objetos de acceso a datos *ADO*, encapsulado en los componentes *ADO*.

Para el diseño de las imágenes que forman parte de la interfase gráfica se usó Adobe Photoshop en su versión 7, teniendo en cuenta sus potencialidades como uno de los mejores sistemas de tratamiento gráfico.

3.4.1 Estado del arte del uso de la tecnología empleada

3.4.1.1 La modelación matemática como herramienta para optimizar problemas económicos - productivos:

Un Modelo es una representación externa y explícita de una parte de la realidad, el cual es visto por individuos que desean usarle para entender, cambiar, manejar y controlar esa parte de la realidad.

La capacidad de modelar y realizar modelos de decisión y análisis es un rasgo esencial entre las muchas aplicaciones reales que van desde los tratamientos médicos de emergencia en las unidades de cuidados intensivos, hasta en los sistemas de control de los comandos militares. Los formalismos y los métodos de inferencia existentes no han sido eficaces en las aplicaciones de tiempo real donde las compensaciones entre la calidad de decisión y manejabilidad computacional son esenciales. En la práctica, un acercamiento eficaz a la modelación de decisión de tiempo dinámico crítico debería proporcionar el apoyo explícito a la modelación de procesos temporales y al manejo de situaciones críticas de tiempo.

Las herramientas para las decisiones tecnológicas tales como los modelos matemáticos, han sido aplicadas a una amplia gama de situaciones en la toma de decisiones dentro de diversas áreas de la gerencia y la economía. En la toma consciente de decisiones bajo incertidumbre, siempre realizamos pronósticos o predicciones. Podríamos pensar que no estamos pronosticando, pero nuestras opciones estarán dirigidas por la anticipación de resultados de nuestras acciones o inacciones.

En la actualidad, los modelos matemáticos son una de las herramientas analíticas más socorridas para la generación de conocimiento en el estudio del crecimiento, reproducción y rendimiento de masas forestales sujetas o no a un régimen de cultivo. La diversidad de modelos en cuanto a su estructura, componentes, la construcción y

propósitos de utilización, es debido a que el crecimiento y la reproducción son procesos complejos y por tanto han justificado gran número de proyectos de investigación [14].

En estos momentos están muy generalizados en el mundo los modelos de optimación lineal de productos múltiples. Es bien sabido que la elección del programa óptimo, es decir, del conjunto de intensidades de las actividades de producción sobre alguna restricción de los recursos o del plan, se traduce en un problema de maximización de una función lineal de muchas variables que satisfaga ciertas restricciones lineales.

Según Kantorovich [14], su amplio y variado uso está determinado por sus propiedades que él mismo expone como:

a. ***Universalidad y flexibilidad***. La estructura del modelo permite diversas formas de su aplicación; puede describir situaciones reales muy diversas para ramas de la economía y niveles de su control muy diferente. Puede considerarse una serie de modelos donde las condiciones y restricciones necesarias se introduzcan paso a paso mientras no se alcance la precisión descriptiva necesaria.

   En los casos más complicados, cuando la hipótesis de la linealidad contradiga significativamente los aspectos específicos del problema y debamos tomar en cuenta insumos y productos no lineales, las decisiones individuales y la información no determinista, el modelo lineal se vuelve un "bloque elemental" y el punto de partida de las generalizaciones.

b. ***Sencillez***. A pesar de su universalidad y buena precisión, el modelo lineal es muy elemental en cuanto a sus medios, que son sobre todo los del álgebra lineal, de modo que aun las personas dotadas de un adiestramiento matemático muy modesto pueden entenderlo y dominarlo. Esto último es muy importante para el uso creativo y no rutinario de los medios analíticos provistos por el modelo.

c. ***Eficiencia de la computación***. La urgencia de la solución de problemas lineales extremos condujo a la elaboración de métodos especiales, muy eficientes, tanto en la URSS (método de mejoramiento sucesivos, método de multiplicadores de resolución) como en los Estados Unidos (el conocido método simplex de G. Dantzig), así como a la producción de una teoría detallada de estos métodos. La estructura algorítmica de los métodos ha permitido escribir después los correspondientes códigos de computadora y ahora las variantes modernas de los métodos de las computadoras modernas pueden resolver rápidamente algunos

problemas con centenares y millares de restricciones, con decenas y cientos de miles de variables.

d. ***Análisis e indicadores cualitativos***. Junto con la solución de planeación óptima, el modelo produce útiles instrumentos de análisis cualitativo de tareas concretas y del problema en conjunto. Esta posibilidad está dada por un sistema de indicadores de las actividades y factores limitantes que se encuentran al mismo tiempo con la solución óptima y de acuerdo con ella. El profesor T. Koopmans llamó "precios sombra" a estos indicadores; yo los llamé "multiplicadores de resolución" porque se utilizan como un recurso auxiliar para el hallazgo de la solución óptima, a la manera de los multiplicadores de Lagrange. Sin embargo, poco después se advirtieron su significado e importancia económicos para el análisis, modo que en el tratamiento económico se han llamado evaluaciones objetivamente determinadas. Estos indicadores tienen el sentido de índices de valor de equivalencia de bienes y factores, determinados en forma intrínseca para un problema dado, y muestran cómo pueden intercambiarse los bienes y factores en las fluctuaciones del estado extremo. Así pues, estas evaluaciones generan un procedimiento objetivo para el cálculo de los precios contables y otros indicadores económicos, y para el análisis de su estructura.

e. ***Concordancia de los medios con los problemas***. Aunque algunas empresas individuales y aun los organismos gubernamentales de estados de economía capitalista han utilizado con éxito estos métodos, su espíritu corresponde más de cerca a los problemas de una economía socialista. La prueba de su eficiencia se encuentra en la aplicación afortunada a varios problemas concretos de la ciencia económica y la investigación de operaciones.

Kantorovich [14] expone la eficiencia de la computación como una de las propiedades principales de estos modelos por la urgencia de su solución. En efecto, la computadora permitió la construcción de modelos matemáticos mucho más complejos que los empleados antes de su existencia; es decir modelos con más variables y más relaciones funcionales entre estas. Pero, para procesarlos, dado que en la mayoría de los casos sus ecuaciones no podían ser resueltas en su forma original por una computadora, los modelos debieron ser alterados con el fin de adaptarlos a métodos numéricos aproximados de resolución, que efectivamente puedan pasarse por la computadora.

3.4.1.2 Las técnicas de programación y el uso de tipos abstractos como elementos esenciales para resolver problemas numéricos de manera eficiente:

Uno de los puntos más cruciales asociados con el diseño de un nuevo sistema involucra la gestión de la complejidad del proceso de diseño. De forma habitual, los buenos diseñadores emplean alguna forma de abstracción como herramienta para tratar esta complejidad. Nuestro uso del término abstracción aquí se refiere a la capacidad intelectual de considerar una entidad aislándola de cualquier ejemplar específico de esa entidad. Por ejemplo, los diseñadores de hardware que pretenden diseñar una computadora, habitualmente se preocupan de la funcionalidad de los circuitos integrados que piensan usar, no del funcionamiento de los transistores que se encuentran en esos circuitos integrados.

El desarrollo de un sistema informático también se ve simplificado en gran medida mediante el uso de la abstracción en el proceso de diseño. En el diseño de software, esto supone especificar la funcionalidad del sistema informático en términos generales de alto nivel. Una vez que puede ser demostrado que esta especificación abstracta del sistema es correcta, es posible añadir más detalle, para finalmente conducir a una descripción detallada de bajo nivel del sistema informático en términos que son directamente implementables usando la sintaxis de un lenguaje de programación dado. En cada paso de este proceso, el diseñador debe verificar que el detalle adicional añadido al diseño del sistema es correcto. La ventaja de este enfoque está en que limita la complejidad con la que se debe tratar en cualquier paso del proceso de diseño. Esto permite al diseñador concentrarse en el conjunto del diseño del sistema sin tener que enfangarse con detalles de implementación.

El uso de la abstracción de datos durante el desarrollo de software permite al diseñador concentrarse en cómo son usados los datos en el sistema para resolver el problema que le ocupa sin tener que preocuparse de cómo los datos son representados y tratados en la memoria de la computadora.

Resulta beneficioso expresar ciertos aspectos del problema en términos de un modelo formal, ya que una vez que el problema se formaliza podemos buscar soluciones mediante un modelo preciso y determinar si ya existe un programa para resolverlo. Aún

en caso de que no exista dicho programa, al menos podemos buscar lo que sea conocido sobre este modelo y usar sus propiedades para ayudarnos a construir una buena solución.

Por ejemplo, algunos problemas numéricos pueden modelarse mediante conceptos matemáticos tales como las matrices y los sistemas de ecuaciones lineales. Los problemas de procesamiento de símbolos y de textos pueden modelarse con cadenas de caracteres y gramáticas formales, como en el caso de los traductores y compiladores.

Emplear una representación concreta durante el diseño puede producir un programa que sea dependiente de una implementación particular de un tipo de datos. Si la implementación del tipo de datos es posteriormente modificada para obtener una representación más eficiente, entonces el programa puede llegar a ser no válido, ya que se diseñó cuando se consideraba la implementación original del tipo de datos. Este tipo de problemas puede ser evitado con el uso de ***tipos abstractos de datos*** en el proceso de diseño.

Un tipo abstracto de datos (TAD) se define como "un modelo matemático de los objetos de datos que constituyen un tipo de datos, así como de las funciones que operan sobre estos objetos" [12]. Es importante apreciar que las operaciones que manipulan los objetos están incluidas en la especificación de un TAD. Por ejemplo, TAD *CONJUNTO* puede ser definido como una colección de elementos que son tratados por operaciones como la *unión*, la *intersección* y la *diferencia* de conjuntos.

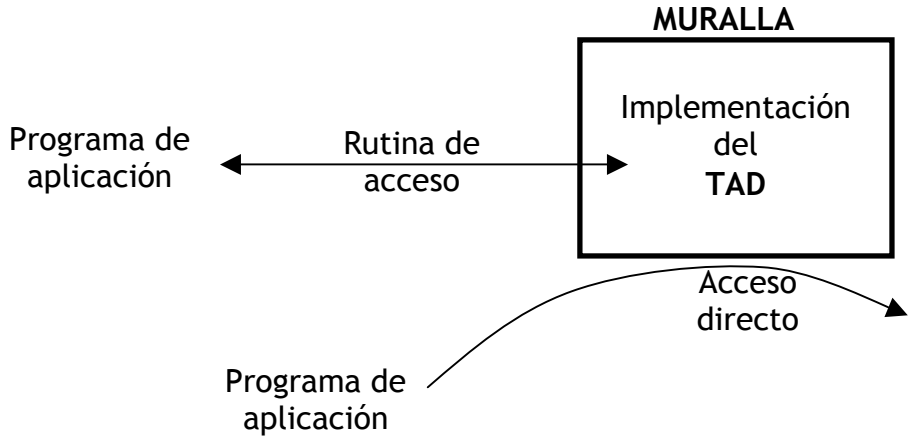
En muchos casos el diseño de un programa informático requerirá de tipos de datos que no se encuentran disponibles en el lenguaje de programación usado para implementar el programa. En estos casos debemos ser capaces de construir los tipos de datos necesarios usando los tipos predefinidos, que a menudo supone la construcción de estructuras de datos realmente complicadas. Los tipos construidos de esta manera se denominan ***tipos de datos definidos por el usuario***.

Dijkstra [5] estudia cómo el proceso de abstracción puede ser usado para reducir la cantidad de complejidad o detalle que debe ser considerado en cualquier momento durante el diseño e implementación de grandes sistemas informáticos.

Una introducción a las técnicas algebraicas de especificación de tipos abstractos de datos se ofrece en Guttag [11]. Goguen et-al. [9] presentan un tratamiento más teórico de esta materia.

Un programa de aplicación debería manipular un tipo de datos en función de su representación lógica más que de su almacenamiento físico. Es decir, la comunicación entre el programa de aplicación y la implementación del tipo de datos debería producirse solamente a través de la interfase que aportan las rutinas de acceso especificadas en el TAD. Esta "separación" u "ocultación" de la representación de un tipo de datos con respecto a las aplicaciones que usan el tipo de datos se conoce como "*encapsulación de datos*". En la terminología de ingenieros de software, la encapsulación de datos se dice que es una forma de ocultación de la información.

Una ventaja que se obtiene de la encapsulación de tipos de datos es la facilidad de modificación. Un tipo de datos encapsulado adecuadamente puede ser modificado sin afectar a los programas de aplicación que usan el tipo de datos. Otra ventaja se refiere a la reutilización. Un tipo de datos debidamente encapsulado puede ser fácilmente reutilizado en otros programas de aplicación que requieran la funcionalidad aportada por el tipo de datos sin tener que conocer cómo fue implementado.



Idealmente una muralla se levanta alrededor de la implementación de un TAD de tal forma que un programa de aplicación solo es capaz de manipular datos por medio del uso de rutinas de acceso. En este caso la manipulación directa de los datos almacenados en la implementación del TAD no está permitida

Figura 3. 3. Representación del encapsulamiento.

En nuestro caso centraremos el análisis sobre los tipos *CONJUNTO*, *MATRTIZ y GRAFO*, por ser los que mayor incidencia tienen en la solución del problema de automatización.

El *CONJUNTO* es una estructura fundamental en las matemáticas que se emplea para agrupar objetos. Los objetos de un conjunto de denominan *elementos* o *miembros* del conjunto. Estos elementos se toman de un *conjunto universal* ***U,*** que contiene todos los posibles elementos de conjuntos. De este modo cualquier conjunto dado consistirá en elementos del conjunto U. Además, todos los miembros de un conjunto dado son únicos; esto es, un conjunto no puede tener dos o más copias del mismo elemento.

A menudo resulta útil considerar estructuras de datos como los conjuntos en los cuales el orden de los elementos es importante. Nos referiremos a una colección de *n* elementos de esta forma como una *n-tupla ordenada*. Concretamente, una *n*-tupla ordenada
$(a_1, a_2,... a_n)$ es una colección de n elementos en la cual a_1 es el primer elemento, a_2 es el segundo elemento, y así sucesivamente. Las 2-tuplas ordenadas se conocen como pares ordenados.

En muchas aplicaciones estamos interesados en especificar una relación entre un conjunto finito de elementos. Debido a su simplicidad, los *GRAFOS* a menudo se emplean para representar tales relaciones. Un *grafo no dirigido* G = (V, E) se compone de dos conjuntos finitos: el *conjunto de nodos* V = {v_1, v_2,...}, que contiene el conjunto de nodos de G; y el *conjunto de aristas* E = {e_1, e_2,...}, que es el conjunto de pares no ordenados de nodos diferentes de G. Cada elemento del conjunto de aristas *e* = (u, v) se conoce como una *arista* y se dice que une los nodos u y v. Si *e* = (u, v) es una arista de G entonces los nodos u y v son adyacentes. Además, se dice que *e* es *incidente en* los nodos u y v. Dado que estamos tratando con pares no ordenados, (u, v) y (v, u) representan la misma arista. Además, como los elementos del par deben ser distintos, los *bucles* –una arista desde un nodo a él mismo- no son posibles. El *grado* de un nodo en un grafo no dirigido viene dado por el número de aristas en un nodo.

Típicamente, un grafo no dirigido se representa por medio de un diagrama, con los nodos mostrados como círculos y las aristas como segmentos. La Figura 3.3 (a) es un diagrama del grafo no dirigido G =(V, E) con conjunto de nodos V ={v_1,v_2,v_3,v_4} y conjunto de aristas E ={e_1=(v_1,v_4), e_2=(v_1,v_2), e_3=(v_1,v_3), e_4=(v_3,v_4)}. En esta figura el grado del nodo v_1 es 3, el grado del nodo v_2 es 1 y el grado de los nodos v_3 y v_4 es 2.

Un *grafo dirigido* G = (V, E) también consiste en un conjunto de nodos y un conjunto de aristas. Sin embargo, en los grafos dirigidos, el conjunto de aristas consiste en pares ordenados de nodos V. En otras palabras, *la dirección* de la arista es importante en los grafos dirigidos. De este modo, la arista e = (u, v) que une los nodos u y v en el grafo dirigido G se dice que es *incidente desde* el nodo u e *incidente hacia* el nodo v. Además, v se dice que es *adyacente hacia* u y u se dice que es *adyacente desde* v. El *grado de salida* de un nodo en un dígrafo viene dado por el número de aristas que son incidentes desde él; mientras que el *grado de entrada* viene dado por el número de aristas incidentes hacia él. El grado de un nodo en un grafo dirigido se define como la suma de los grados de entrada y salida del nodo. Como los grafos no dirigidos, los grafos dirigidos también pueden ser representados por medio de un diagrama. En este caso, las aristas se dibujan con flechas para mostrar la dirección de la arista. La Figura 3.3 (b) es un diagrama del grafo dirigido G =(V, E) con conjunto de nodos V = {v_1,v_2,v_3} y conjunto de aristas E = {e_1=(v_1,v_2), e_2=(v_2,v_1), e_3=(v_3,v_1), e_4=(v_3,v_2)}. El grado de entrada de v_1 es 2, de v_2 es 2 y de v_3 es 0; mientras que el grado de salida de v_1 es 1, de

v_2 es 1 y de v_3 es 2. así el grado de v_1 y v_2 es 3 y el de v_3 es 2. Cuando se usan grafos dirigidos o no dirigidos para modelar ciertas relaciones, a menudo resulta útil asociar un peso a cada arista de un grafo. Usaremos w(u, v) para denotar el peso asociado a la arista (u, v). Una asignación de pesos a un grafo G = (V, E) viene dada por una aplicación w: $\{(u, v) \in E\} \rightarrow \Re$. Nos referiremos a los grafos que tienen una asignación como esta como *grafos valorados*.

Muchos autores remontan el origen de la teoría de grafos al famoso problema de Los *Puentes de Königdberg* resuelto por Leonhard Euler en 1736. Otro famoso problema resuelto usando técnicas de teorías de grafos resulta memorable porque una computadora fue instrumento de su solución. En el problema de los cuatro colores se nos pide determinar si los países de cualquier mapa en el plano (o en la esfera) pueden ser coloreados con cuatro o menos colores, de tal forma que todos los países con fronteras comunes (que no sean un único punto) tengan colores diferentes. La prueba suministrada por Appel et al. [1] (para una exposición de alto nivel de este trabajo), trataba los países como nodos de un grafo que estaban conectados por una arista si compartían una frontera. Su prueba es combinatorialmente complicada, requiriendo la comprobación de un gran número de casos difíciles. Esta comprobación, que hubiera sido insondable "manualmente", fue completada usando 1200 horas de tiempo de cómputo.

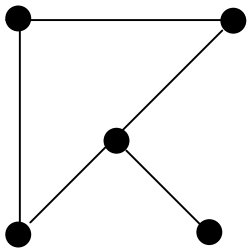
Figura 3. 4 (a) Un grafo no dirigido que contiene 4 nodos y 4 aristas. (b) Un grafo dirigido que contiene 3 nodos y 4 aristas.

Para resolver un problema de grafos utilizando una computadora, debemos ser capaces de almacenar grafos en la memoria de la computadora. Así, necesitamos considerar la representación de un grafo como una estructura de datos. Las dos formas más comunes de hacer esto según lo planteado por Heileman [12] consisten en utilizar o bien una

MATRIZ de adyacencia, o bien una *LISTA* de adyacencia. Una representación con una matriz de adyacencia de un grafo G =(V, E) se almacena en un array bidimensional A[1.. |V|, 1.. |V|], donde

$$A[i,j] = \begin{cases} 1, & \text{si } (i,j) \in E \\ 0, & \text{si } (i,j) \notin E \end{cases}$$

Un grafo ejemplo se muestra en la Figura 3.4 (a) y su correspondiente matriz de adyacencia se muestra en la Figura 3.4 (b). Esta representación se extiende fácilmente a grafos valorados reemplazando cada componente <<i>> de la matriz de adyacencia correspondiente a una arista (u, v) con el peso w (u, v) asociado con esa arista. Las matrices de adyacencia de los grafos no dirigidos poseen cierto número de propiedades interesantes. Como la autoincidencia no está permitida en los grafos no dirigidos, la diagonal principal solo contiene ceros. Además, como (*i, j*) y (*j, i*) representan la misma arista en un grafo no dirigido, la matriz de adyacencia de un grafo no dirigido es simétrica con respecto a su diagonal principal.



(a)

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	0	0	1
3	0	0	0	0	1
4	1	0	0	0	1
5	0	1	1	1	0

(b)

Figura 3. 5 (a) Un grafo no dirigido de ejemplo y su representación con una matriz de adyacencia. (b) Una matriz de adyacencia.

El uso de tipos abstractos de datos para modelar problemas numéricos trae consigo la complicación de tareas de programación, que en la mayoría de los casos suelen ser complejas. En este contexto, la programación orientada a objetos (POO), surge como el mejor paradigma disponible para enfrentar estas tareas.

La programación orientada a objetos es la expresión de uno de los más avanzados paradigmas en el campo de la programación, y es, al mismo tiempo, el resultado de la evolución experimentada por los paradigmas anteriores.

A diferencia de otros paradigmas de programación, que intentan, al abordar un problema, representarlo o modelarlo empleando entidades cercanas a la computadora (arreglos, subrutinas, módulos) la programación orientada a objetos se propone emplear entidades lo más cercanas posibles a la realidad. En Peterson [17] se presenta una colección de artículos que abordan los conceptos de la programación orientada a objetos.

El cómputo de un sistema orientado a objetos supone la manipulación de objetos de ciertas clases. Una clase es en realidad un medio de empaquetar un TAD. Esto es, el concepto de clase permite encapsular conjuntamente como una única entidad a los elementos, así como a las rutinas de acceso de la implementación de un TAD. La idea de incorporar TAD en un lenguaje de programación se remonta a la construcción *class* encontrada en el lenguaje SIMULA 67 [3]. Stroustrup [21] afirma que el lenguaje C++ fue inventado originalmente porque "quería escribir algunas simulaciones conducidas por sucesos para las cuales SIMULA 67 habría sido ideal, excepto por consideraciones de eficiencia".

Una clase contiene toda la información necesaria para construir ejemplares individuales de ella misma; estos ejemplares se conocen como *objetos*. Si se selecciona a los objetos para representar a los objetos del mundo real, se puede obtener una correspondencia más natural entre el dominio del problema y el dominio de la aplicación.

El diseño de programas orientados a objetos se logra mejor si se analizan las clases de objetos en el sistema físico. Este enfoque de diseño consiste en el refinamiento sucesivo de las descripciones e interacciones de estos objetos. La estructura de clases ofrecida por los lenguajes de programación orientados a objetos apoya este enfoque permitiendo que los objetos del sistema físico sean modelados con los mismos objetos en el sistema software. Además estas clases pueden extenderse o combinarse fácilmente, posibilitando que formen componentes más complejas.

La posibilidad de estructurar un sistema informático conforme a clases ofrece un medio útil para descomponer los componentes del sistema; sólo esto ya conduce a un sistema más manejable. Sin embargo, la posibilidad de extensión así como la reutilización de los componentes software se mejora si se incorpora el concepto de *herencia*. La herencia es el medio por el cual los objetos de una clase pueden acceder a variables y funciones miembro contenidas en una clase previamente definida, sin tener que volver realizar esas definiciones. Esto nos dará la posibilidad de crear una nueva clase que sea una extensión o especialización de una clase existente. La relación entre clases en un programa orientado a objetos se ilustra a menudo como un *grafo de herencia*.

Una clase derivada hereda variables miembro, variables de clase y funciones miembro de su clase base. La clase derivada puede también añadir nuevas variables miembro, variables de clase, o funciones miembros que sean necesarias para sus operaciones especializadas. Adicionalmente, una clase derivada puede redefinir cualquier función miembro aportada por la clase base simplemente suministrando una nueva función miembro que tenga el mismo nombre que la antigua función miembro en la clase base.

Uno de los aspectos más difíciles del diseño orientado a objetos consiste en determinar la organización de las clases en la arquitectura del sistema software. Hay muchas formas diferentes en las que las clases y los objetos pueden interactuar. Definir relaciones entre las entidades que pueden existir en un sistema orientado a objetos hace más fácil entender, estudiar y modificar el sistema. Consideremos las siguientes relaciones:

1. Relación ES-UN(A): ésta es una relación de especialización que se utiliza para indicar que una clase es una variante de otra clase. Afirmar que “la clase B ES-UNA clase A” indica que las principales características de la clase B se heredan de la clase A.
2. Relación TIENE-UN(A): ésta es una relación de contenido que se utiliza para indicar que una clase u objeto es parte de otra clase u objeto. Mientras que la relación ES-UNA solo puede ser usada para definir relaciones entre clases; TIENE-UN(A) puede usarse para definir una relación entre clases, entre un objeto y una clase o entre objetos. Afirmar que “la clase B TIENE-UN objeto A” indica que el objeto A es una componente de la clase B. esto es, la clase A se utiliza como una pieza en la construcción de la clase B.

3. Relación USA-UN(A): ésta es una relación de uso que indica que una función miembro de una clase acepta, y por tanto usa, un objeto de alguna otra clase como parámetro. Por ejemplo, afirmar que "la clase B USA-UNA clase A" indica que los objetos de la clase B utilizan las facilidades ofrecidas por los objetos de la clase A, no que la clase A se emplee como pieza en la construcción de la clase B.

La motivación de la herencia se sigue de la observación de que en muchos casos un nuevo desarrollo de software evoluciona a partir de desarrollos de esfuerzos anteriores. Este hecho, durante mucho tiempo ignorado en el diseño clásico de software, es de vital importancia en el diseño orientado a objetos. Usando herencia, los programadores son capaces de evitar la costosa y proclive a errores tarea de rescribir grandes cantidades de código que realice casi las mismas tareas que alguna pieza de código ya existente. Por tanto, por medio del empleo de la herencia, una adición a un sistema a menudo supone solo eso, adición y no modificación.

Cuando un sistema informático determina cómo debe ser realizada una operación particular sobre un objeto, se dice que *vincula* al objeto una implementación específica de esa operación. Si el sistema decide en tiempo de compilación qué implementación de la operación va a utilizar, realiza una *vinculación estática*. Si esta elección se hace en tiempo de ejecución, entonces el sistema realiza una vinculación dinámica.

Otra característica de los lenguajes orientados a objetos que se apoya en la vinculación dinámica es el *polimorfismo*. El polimorfismo se define como la posibilidad de asumir varias formas. En la programación orientada a objetos este término es utilizado para denotar la posibilidad de que un único mensaje pueda referirse en tiempo de ejecución a objetos de distintas clases.

La capacidad para usar la abstracción de datos es fundamental en cualquier enfoque de diseño. Por tanto, la encapsulación de datos y la abstracción de datos ofrecidas por los lenguajes orientados a objetos nos permiten desarrollar TAD's que ejecutan conjuntos de operaciones bien definidos, independientemente de una aplicación particular.

3.4.1.3 El Proceso Unificado de Desarrollo de Software como paradigma en la construcción de Sistemas

Un proceso define quien esta haciendo qué, cuándo, y cómo alcanzar un determinado objetivo. En la ingeniería del software el objetivo es construir un producto software o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia, reduce el riesgo y hace el proyecto más predecible. El efecto global es el fomento de una visión y una cultura comunes.

Un proceso de desarrollo de software debería también ser capaz de evolucionar durante muchos años. Durante esta evolución debería limitar su alcance, en un momento del tiempo dado, a las realidades que permitan las tecnologías, herramientas, personas y patrones de organización.

- ✓ Tecnologías: El proceso debe construirse sobre las tecnologías -lenguajes de programación, sistemas operativos, computadores, estructuras de red, entornos de desarrollo, etc.- disponibles en el momento en que se va a emplear el proceso. Por ejemplo, hace veinte años el modelado visual no era realmente de uso general. Era demasiado caro. En aquellos tiempos, un creador de un proceso prácticamente tenía que asumir que se usarían diagramas hechos a mano. Esa suposición limitaba mucho el grado en el cual el creador del proceso podía establecer el modelado dentro del proceso.
- ✓ Herramientas. Los procesos y las herramientas deben desarrollarse en paralelo. Las herramientas son esenciales en el proceso. Dicho de otra forma, un proceso ampliamente utilizado puede soportar la inversión necesaria para crear las herramientas que lo soporten.
- ✓ Personas. Un creador del proceso debe limitar el conjunto de habilidades necesarias para trabajar en el proceso a las habilidades que los desarrolladores actuales poseen, o apuntar aquellas que los desarrolladores puedan obtener rápidamente. Hoy es posible empotrar en herramientas software técnicas que antes requerían amplios conocimientos, como la comprobación de la consistencia en los diagramas del modelo.
- ✓ Patrones de organización. Aunque los desarrolladores de software no pueden ser expertos tan independientes como los músicos de una orquesta, están muy lejos de

los trabajadores autómatas en los cuales Frederick W. Taylor basó su "dirección científica" hace cien años.

La tendencia actual en el software lleva a la construcción de sistemas más grandes y más complejos. Esto es debido en parte al hecho de que los computadores son más potentes cada año, y los usuarios, por tanto, esperan más de ellos. Esta tendencia también se ha visto afectada por el uso creciente de Internet para el intercambio de todo tipo de información –de texto sin formato a texto con formato, fotos, diagramas y multimedia. Nuestro apetito de software aun más sofisticado crece a medida que vemos cómo pueden mejorarse los productos de una versión a otra. Queremos un software que este mejor adaptado a nuestras necesidades, pero esto, a su vez, simplemente hace el software más complejo. En breve, querremos más. También lo queremos más rápido. El tiempo de salida al mercado es otro conductor importante.

Conseguirlo, sin embargo, es difícil. Nuestra demanda de software potente y complejo no se corresponde con cómo se desarrolla el software. Hoy la mayoría de la gente desarrolla software mediante los mismos métodos que llevan utilizándose desde hace 25 años. Esto es un problema. A menos que renovemos nuestros métodos, no podremos cumplir con el objetivo de desarrollar el software complejo que se necesita actualmente.

El problema del software se reduce a la dificultad que afrontan los desarrolladores para coordinar las múltiples cadenas de trabajo de un gran proyecto de software. La comunidad de desarrolladores necesita una forma coordinada de trabajar. Necesita un proceso que integre las múltiples facetas del desarrollo. Necesita un método común, un proceso que:

- ✓ Proporcione una guía para ordenar las actividades de un equipo.
- ✓ Dirija las tareas de cada desarrollador por separado y del equipo como un todo.
- ✓ Especifique los artefactos que deben desarrollarse.
- ✓ Ofrezca criterios para el control y la medición de los productos y actividades del proyecto.

El Proceso Unificado de Desarrollo –el resultado de más de 30 años de experiencia- es una solución al problema del software.

En primer lugar, el Proceso Unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software (véase la Figura 3.5). Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema software en construcción esta formado por componentes software interconectados a través de interfases bien definidas.

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (*UML*) para preparar todos los esquemas de un sistema software. De hecho, *UML* es una parte esencial del Proceso Unificado –sus desarrollos fueron paralelos.

No obstante, los verdaderos aspectos definitorios del Proceso Unificado se resumen en tres frases clave –dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental. Esto es lo que hace único al Proceso Unificado.

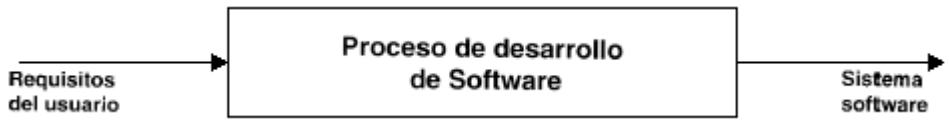


Figura 3. 6. Un proceso de desarrollo de software.

El Proceso Unificado está dirigido por casos de uso

Un sistema software ve la luz para dar servicio a sus usuarios. Por tanto, para construir un sistema con éxito debemos conocer lo que sus futuros usuarios necesitan y desean.

El término usuario no solo hace referencia a usuarios humanos sino a otros sistemas. En este sentido, el término usuario representa alguien o algo (como otro sistema fuera del sistema en consideración) que interactúa con el sistema que estamos desarrollando. Un

ejemplo de interacción sería una persona que utiliza un cajero automático. Él (o ella) inserta la tarjeta de plástico, responde a las preguntas que le hace la máquina en su pantalla, y recibe una suma de dinero. En respuesta a la tarjeta del usuario y a sus contestaciones, el sistema lleva a cabo una secuencia de acciones que proporcionan al usuario un resultado importante, en este caso, la retirada del efectivo.

Una interacción de este tipo es un caso de uso. Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso, el cual describe la funcionalidad total del sistema. Puede decirse que una especificación funcional contesta a la pregunta: ¿Qué debe hacer el sistema? La estrategia de los casos de uso puede describirse añadiendo tres palabras al final de esta pregunta: ¿...para cada usuario? Estas tres palabras albergan una implicación importante. Nos fuerzan a pensar en términos de importancia para el usuario y no solo en términos de funciones que sería bueno tener. Sin embargo, los casos de uso no son solo una herramienta para especificar los requisitos de un sistema. También guían su diseño, implementación, y prueba; esto es, guían el proceso de desarrollo. Basándose en el modelo de casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que llevan a cabo los casos de uso.

Los desarrolladores revisan cada uno de los sucesivos modelos para que sean conformes al modelo de casos de uso. Los ingenieros de prueba prueban la implementación para garantizar que los componentes del modelo de implementación implementan correctamente los casos de uso. De este modo, los casos de uso no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor. Dirigido por casos de uso quiere decir que el proceso de desarrollo sigue un hilo –avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso se especifican, se diseñan, y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba.

Aunque es cierto que los casos de uso guían el proceso, no se desarrollan aisladamente. Se desarrollan a la vez que la arquitectura del sistema. Es decir, los casos de uso guían la arquitectura del sistema y la arquitectura del sistema influye en la selección de los

casos de uso. Por tanto, tanto la arquitectura del sistema como los casos de uso maduran según avanza el ciclo de desarrollo.

El Proceso Unificado está centrado en la arquitectura

El papel de la arquitectura software es parecido al papel que juega la arquitectura en la construcción de edificios. El edificio se contempla desde varios puntos de vista: estructura, servicios, conducción de la calefacción, fontanería, electricidad, etc. Esto permite a un constructor ver una imagen completa antes de que comience la construcción. Análogamente, la arquitectura en un sistema software se describe mediante diferentes vistas del sistema en construcción.

El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y los inversores, y se refleja en los casos de uso. Sin embargo, también se ve influida por muchos otros factores, como la plataforma en la que tiene que funcionar el software (arquitectura hardware, sistema operativo, sistema de gestión de base de datos, protocolos para comunicaciones en red), los bloques de construcción reutilizables de que se dispone (por ejemplo, un marco de trabajo para interfases gráficas de usuario), consideraciones de implantación, sistemas heredados, y requisitos no funcionales (por ejemplo, rendimiento, fiabilidad). La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado. Debido a que lo que es significativo depende en parte de una valoración, que a su vez, se adquiere con la experiencia, el valor de una arquitectura depende de las personas que se hayan responsabilizado de su creación. No obstante, el proceso ayuda al arquitecto a centrarse en los objetivos adecuados, como la comprensibilidad, la capacidad de adaptación al cambio, y la reutilización.

¿Cómo se relacionan los casos de uso y la arquitectura? Cada producto tiene tanto una función como una forma. Ninguna es suficiente por sí misma. Estas dos fuerzas deben equilibrarse para obtener un producto con éxito. En esta situación, la función corresponde a los casos de uso y la forma a la arquitectura. Debe haber interacción entre los casos de uso y la arquitectura. Es un problema del tipo "el huevo y la gallina". Por un lado, los casos de uso deben encajar en la arquitectura cuando se llevan a cabo. Por

otro lado, la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, ahora y en el futuro. En realidad, tanto la arquitectura como los casos de uso deben evolucionar en paralelo.

Por tanto, los arquitectos moldean el sistema para darle una forma. Es esta forma, la arquitectura, la que debe diseñarse para permitir que el sistema evolucione, no solo en su desarrollo inicial, sino también a lo largo de las futuras generaciones. Para encontrar esa forma, los arquitectos deben trabajar sobre la comprensión general de las funciones clave, es decir, sobre los casos de uso claves del sistema. Estos casos de uso clave pueden suponer solamente entre el 5 y el l0 por ciento de todos los casos de uso, pero son los significativos, los que constituyen las funciones fundamentales del sistema. De manera resumida, podemos decir que el arquitecto:

- ✓ Crea un esquema en borrador de la arquitectura, comenzando por la parte de la arquitectura que no es especifica de los casos de uso (por ejemplo, la plataforma). Aunque esta parte de la arquitectura es independiente de los casos de uso, el arquitecto debe poseer una comprensión general de los casos de uso antes de comenzar la creación del esquema arquitectónico.
- ✓ A continuación, el arquitecto trabaja con un subconjunto de los casos de uso especificados, con aquellos que le presenten las funciones clave del sistema en desarrollo. Cada caso de uso seleccionado se especifica en detalle y se realiza en términos de subsistemas, clases, y componentes.
- ✓ A medida que los casos de uso se especifican y maduran, se descubre más de la arquitectura. Esto, a su vez, lleva a la maduración de más casos de uso.

Este proceso continúa hasta que se considere que la arquitectura es estable.

El Proceso Unificado es iterativo e incremental

El desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o mini-proyectos. Cada mini-proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas; esto es, deben seleccionarse y ejecutarse de una forma planificada. Es por esto por lo que son mini-proyectos.

Los desarrolladores basan la selección de lo que se implementará en una iteración de dos factores. En primer lugar, la iteración trata un grupo de casos de uso que juntos amplían la utilidad del producto desarrollado hasta ahora. En segundo lugar, la iteración trata los riesgos más importantes. Las iteraciones sucesivas se construyen sobre los artefactos de desarrollo tal como quedaron al final de la última iteración. AI ser mini-proyectos, comienzan con los casos de uso y continúan a través del trabajo de desarrollo subsiguiente –análisis, diseño, implementación y prueba-, que termina convirtiendo en código ejecutable los casos de uso que se desarrollaban en la iteración. Por supuesto, un incremento no necesariamente es aditivo. Especialmente en las primeras fases del ciclo de vida, los desarrolladores pueden tener que reemplazar un diseño superficial por uno más detallado o sofisticado. En fases posteriores, los incrementos son típicamente aditivos.

En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componentes, y verifican que los componentes satisfacen los casos de uso. Si una iteración cumple con sus objetivos –como suele suceder- el desarrollo continúa con la siguiente iteración. Cuando una iteración no cumple sus objetivos, los desarrolladores deben revisar sus decisiones previas y probar con un nuevo enfoque.

Para alcanzar el mayor grado de economía en el desarrollo, un equipo de proyecto intentará seleccionar sólo las iteraciones requeridas para lograr el objetivo del proyecto. Intentará secuenciar las iteraciones en un orden lógico. Un proyecto con éxito se ejecutara de una forma directa, solo con pequeñas desviaciones del curso que los desarrolladores planificaron inicialmente. Por supuesto, en la medida en que se añadan iteraciones o se altere el orden de las mismas por problemas inesperados, el proceso de desarrollo consumirá más esfuerzo y tiempo. Uno de los objetivos de la reducción del riesgo es minimizar los problemas inesperados.

Son muchos los beneficios de un proceso iterativo controlado:

- ✓ La iteración controlada reduce el coste del riesgo a los costes de un solo incremento. Si los desarrolladores tienen que repetir la iteración, la organización solo pierde el esfuerzo mal empleado de la iteración, no el valor del producto entero.

- ✓ La iteración controlada reduce el riesgo de no sacar al mercado el producto en el calendario previsto. Mediante la identificación de riesgos en fases tempranas del desarrollo, el tiempo que se gasta en resolverlos se emplea al principio de la planificación, cuando la gente está menos presionada por cumplir los plazos. En el método "tradicional", en el cual los problemas complicados se revelan por primera vez en la prueba del sistema, el tiempo necesario para resolverlos normalmente es mayor que el tiempo que queda en la planificación, y casi siempre obliga a retrasar la entrega.
- ✓ La iteración controlada acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan de manera más eficiente para obtener resultados claros a corto plazo, en lugar de tener un calendario largo, que se prolonga eternamente.
- ✓ La iteración controlada reconoce una realidad que a menudo se ignora –que las necesidades del usuario y sus correspondientes requisitos no pueden definirse completamente al principio. Típicamente, se refinan en iteraciones sucesivas. Esta forma de operar hace más fácil la adaptación a los requisitos cambiantes.

Estos conceptos –los de desarrollo dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental– son de igual importancia. La arquitectura proporciona la estructura sobre la cual guiar las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración. La eliminación de una de las tres ideas reduciría drásticamente el valor del Proceso Unificado.

3.4.2 Caracterización y justificación del soporte de Base de Datos utilizado

Para implementar la base de datos se escogió el SGBD MS Access en su versión 10 (XP). MS Access surgió en su versión 1.0 como un sistema limitado a gestionar bases de datos por debajo de 128 MB, sin capacidad de control de transacciones ni de integridad referencial, con una barra de herramientas estática y un muy lento desempeño.

A la altura de la versión 4. 0 ya incorporaba procesamiento de transacciones, integridad referencial, capacidad de replicación, mejor administración de concurrencias, mayor velocidad y estabilidad en su funcionamiento. Sus posibilidades para desarrollar

automatización de tareas evolucionó desde simples macros (lotes de comandos) hasta un entorno de desarrollo *VBA* como lenguaje de cuarta generación con plenas potencialidades, incluyendo soporte para tecnología *ActiveX*.

Con Access 95 y 97 los desarrolladores obtuvieron la posibilidad de crear múltiples instancias de los formularios. Access 97 también habilitó el uso de páginas ASP (Active Server Pages), extendiendo las posibilidades de desarrollo a las aplicaciones Web.

Con Access 2000 se produce un cambio revolucionario respecto al 97. Se incluye las páginas HTLM como una clase nueva de objetos no encapsulados en el contenedor pero que ofrece una variante altamente productiva para visualizar contenido Web dinámico. Se reorganiza el entorno de desarrollo acorde a la implementación *VBA* 6.0 y aparece el concepto de proyecto Access con un enfoque cliente-servidor total.

Access 2002 es más bien una herramienta evolucionada de 2000 con un ambiente más amigable y mayor eficiencia con mejoras en algunas de sus principales características.

De manera general Access funciona por una parte como un programa que consta de una fuerte variedad y cantidad de asistentes, y por otra como un entorno de desarrollo de programas de aplicaciones orientada a la gestión de bases de datos.

Las tareas ejecutadas con los asistentes son traducidas en forma de cláusulas *SQL* y pasadas al *Jet Engine*, que las ejecuta. El Jet es un componente ActiveX que funciona como un driver de bases de datos. En el tránsito al Jet la orden es preprocesada por un espacio de trabajo *ODBC* u *OLE DB*.

El entorno de desarrollo consta de dos objetos base. El objeto *Engine* está en el tope de la jerarquía de los modelos de acceso a datos *DAO/RDO*, para programar tareas de gestión de datos, y utiliza el espacio de trabajo *ODBC* para llegar al Jet. El objeto *Application* encabeza la jerarquía del modelo de objetos de la *API* del programa, para tareas de construcción de interfases fundamentalmente. El *VBA* suministra funcionalidad ActiveX para el resto de las necesidades que incluyen el uso de otros modelos de datos como *ADO*.

Las características principales de MS Access se pueden resumir en los puntos siguientes:

- ✓ Concepto de contenedor de base de datos, todos los objetos de datos y de tareas automatizadas encapsuladas en la misma estructura física.
- ✓ Modelos de acceso a datos *DAO/RDO* para acceso a datos locales y remotos.
- ✓ Entorno *VBA* para desarrollar aplicaciones con soporte para componentes ActiveX.
- ✓ Construcción de consultas mediante asistente QBE (Query By Example) que suministra la cláusula *SQL* de consulta en construcción.
- ✓ Modelo de seguridad a nivel de base de datos y de usuario.
- ✓ Asistentes para la creación de los objetos de base de datos y de aplicación.
- ✓ Proyectos Access para aplicaciones cliente-servidor.
- ✓ Administrador de replicas.
- ✓ Administrador de concurrencias.
- ✓ Control de transacciones.
- ✓ Semi-compilación de objetos de aplicación.

3.4.2.1 Caracterización del enfoque de diseño de bases de datos utilizado

El modelo de Bases de datos Relacional se soporta teórica y conceptualmente en elementos de la teoría de conjuntos. El concepto de relación es el más importante dentro del Modelo Relacional de diseño de bases de datos. Este modelo es el más utilizado en todo el mundo para diseñar bases de datos, ya que mediante él se pueden modelar infinidad de entornos informativos. Los principales Sistemas de gestión de Bases de Datos (SGBD) implementan este modelo. Es preciso comentar que al implementar un diseño de datos en un SGBD específico, las relaciones definidas se convierten en tablas físicas, es decir, las bases de datos relacionales devienen en conjuntos de tablas que almacenan registros de datos.

En un momento del desarrollo de las bases de datos se intentó crear un Modelo de Bases de Datos Orientado a Objetos, para transferir al diseño de bases de datos las ventajas del modelo de diseño orientado a objetos. Sin embargo, la consistencia del modelo relacional, su capacidad para generar diseños sencillos y eficaces, hizo imposible renunciar a sus principios. Actualmente existe un Modelo Objeto-Relacional que trata de sacar lo mejor de cada modelo. Algunas implementaciones del lenguaje *SQL*, como

la de Oracle 8 y superiores, implementan cláusulas para crear diseños de datos acorde al Modelo Objeto-Relacional. También existen lenguajes de modelación como el *UML*, que se pueden utilizar para resolver el modelo conceptual y obtener el diagrama de la base de datos con elementos Objeto-Relacionales. Pero el Modelo Relacional sigue siendo el más ampliamente usado y su efectividad no ha disminuido a pesar de los nuevos requerimientos surgidos para el diseño de bases de datos como la inclusión de elementos vectoriales.

Existen diferentes vías para utilizar el Modelo Relacional en el diseño de una base de datos. El Modelo Entidad-Relación (MER), propuesto en 1976 con una amplia aceptación, es el modelo escogido para presentar el diseño de la base de datos de APROV PLUS.

El MER opera con dos conceptos fundamentales: Entidad y Relación. Una entidad es la ocurrencia de un conjunto de entidades. En rigor, un conjunto de entidades es una relación matemática, es decir, un conjunto de n-uplas ordenadas. Una relación es una ocurrencia de un conjunto de relaciones. También es en rigor una relación matemática, solo que en el MER expresa el vínculo entre entidades de conjuntos previamente definidos.

Al aplicar el MER para diseñar la base de datos de APROV PLUS se siguen los pasos generales para el diseño de una base de datos. Estos son:

1. Obtención del Modelo Conceptual.
2. Obtención del Modelo Lógico.
3. Control de la calidad de los datos.
4. Implementación del diseño de la base de datos.

El modelo conceptual comprende la definición de los conjuntos de entidades, que llamaremos simplemente Entidades; de los conjuntos de relaciones, que llamaremos Relaciones; y los atributos de las Entidades y sus Relaciones.

Se presentaron varios casos de relaciones que se obtienen por abstracción de atributos comunes de varias entidades. En estos casos se aplica una de las extensiones al MER, que consiste en operaciones de Generalización - Especialización. Esto es definir

entidades generalizadas con los atributos que son comunes a varias entidades y dejar en estas últimas los atributos que son específicos de ellas, es decir, como entidades especializadas.

Aparece otro caso de entidades relacionadas con cardinalidad **n a n** donde esta relación tiene relación con otra entidad. En este caso se aplica otra extensión del MER que consiste en la Agregación de entidades, es decir, una nueva entidad como resultado de la relación **n a n** entre las dos primeras.

La representación diagrámica es el último paso del Modelo Conceptual. Utilizando los elementos de representación gráfica del MER se obtiene el Diagrama Entidad-Relación (DER), que es la síntesis de todos los elementos definidos en el modelo conceptual. El DER es la forma más completa y efectiva de presentar el diseño de la base de datos.

El Modelo Lógico se obtiene mediante el lenguaje de implementación Lógico del Modelo Relacional. Esto significa obtener las tablas lógicas que derivan de las entidades y sus relaciones y que está más cerca de la forma que realmente tendrá la base de datos una vez implementada.

Las tablas lógicas se obtienen directamente del DER siguiendo un procedimiento de ocho pasos. Otra herramienta utilizada es la elaboración de las Tarjetas de Objetos. Para cada tabla se elabora una tarjeta de objeto que contiene la mayor cantidad de información sobre la definición de las tablas. Incluye el nombre definitivo de la tabla y sus atributos, los tipos de datos asignados a los atributos, las restricciones establecidas para su definición, entre otras cosas.

Los elementos del Modelo Lógico unido al DER suministran la información necesaria para implementar la base de datos, o sea para crearla físicamente.
El control de la calidad de los datos se realiza mediante el proceso de Normalización de los datos, que implica chequear si las tablas lógicas cumplen con ciertas normas establecidas. Las normas son conocidas como Formas Normales (FN). Las tablas se chequean empezando por la Primera Forma Normal (1FN), y luego a través del orden siguiente:

1FN → 2FN → 3FN → FN de Boyce-Codd → 4FN → 5FN
2FN significa Segunda Forma Normal, y así sucesivamente.

Dependiendo de la complejidad del diseño no siempre es necesario llegar hasta 5FN para obtener un diseño óptimo. La Normalización de los datos ayuda a evitar problemas relacionados con la redundancia de datos, la pérdida de datos por error durante operaciones de actualización y la ambigüedad en relaciones múltiples entre entidades.

3.4.3 Caracterización y justificación del lenguaje de programación utilizado

El lenguaje usado para el diseño de APROV PLUS fue la implementación de Borland International del Object Pascal, que a su vez se deriva del lenguaje Pascal.

El lenguaje Pascal fue diseñado en los años 70 por Niklaus Wirth, profesor el Instituto Politécnico Federal de Zurich. Este lenguaje, pensado en su origen para la enseñanza de la programación, ha sido adaptado a otros muchos propósitos, lo que permite emplear extensamente este lenguaje también para la programación práctica.

En años posteriores fue desarrollado el Object Pascal por Apple Computer con la asesoría del diseñador de Pascal, el propio Niklaus Wirth.

La compañía Borland Internacional creó una implementación de este lenguaje para incorporarlo a uno de los entornos de desarrollo más versátiles que existe en la actualidad, Borland Delphi, el cual constituye el resultado de la evolución de un entorno que se denominó Turbo Pascal.

Esta compañía tomó el Pascal original, hizo algunos cambios y lanzó el Turbo Pascal. Tuvo varias versiones, muchas de ellas utilizadas en nuestro país durante muchos años. Las primeras corrían sobre el Sistema Operativo DOS, las últimas ya eran para el Sistema Operativo Windows. Se decide la Borland por dar un salto de calidad y adopta el Object Pascal con algunas variantes y lanza al mercado el Delphi, con un entorno que permite hacer aplicaciones Windows de una manera relativamente sencilla. También Delphi ha tenido varias versiones, aunque el lenguaje ha sufrido pocas alteraciones. De una versión a otra lo que ofrecen es una mayor cantidad de recursos de programación de manera que no haya que escribir mucho código para poder hacer un sistema determinado.

Es importante no confundir Delphi con un lenguaje de programación. Delphi es un "Entorno de Desarrollo o de Programación", de tipo RAD (Rapid application developer), que soporta el Object Pascal como lenguaje de programación.

El secreto de Delphi está en la sencillez. Comparte la filosofía de programar en un entorno totalmente visual y en el lenguaje Object Pascal (Pascal orientado a objetos), que permite hacer aplicaciones, ponerlas a punto y obtener un ejecutable de manera que después se pueda correr de forma independiente. Algunas de sus propiedades originales, que atrajeron a los usuarios, fueron su enfoque orientado a objetos y basado en formularios, su compilador extremadamente rápido, su gran soporte para bases de datos, su estrecha integración con la programación para Windows y su tecnología de componentes, pero el elemento más importante era el lenguaje Pascal orientado a objetos.

Delphi 2 era incluso mejor. Entre sus propiedades añadidas más importantes estaban las siguientes: El Multi Record Object y la cuadricula para bases de datos mejorada, el soporte para Automatización OLE y el tipo de datos variant, el soporte e integración totales de Windows 95, el tipo de datos de cadena larga y la herencia de formulario visual.

Delphi 3 añadió la tecnología Code Insight, el soporte de depuración DLL, las plantillas de componentes, el Teechart, el Decisión Cube, la tecnología WebBroker, los paquetes de componentes, los ActiveForms y una sorprendente integración con COM, gracias a las interfases.

Delphi 4 nos trajo el editor AppBrowser, nuevas propiedades de Windows 98, mejor soporte OLE y COM, componentes de bases de datos ampliados y muchas mas clases principales de la VCL añadidas, como el soporte para acoplamiento, restricción y anclaje de los controles.

Delphi 5 añadió a este cuadro muchas mejoras en el IDE como el soporte ampliado para bases de datos (con conjuntos de datos específicos de *ADO* e InterBase), una versión mejorada de MIDAS con soporte para Internet, la herramienta de control de versiones TeamSource, capacidades de traducción, el concepto de marcos y nuevos componentes.

Delphi 6 añadió a todas estas propiedades, el soporte para el desarrollo multiplataforma con la nueva biblioteca de componentes para multiplataforma (CLX), una biblioteca en tiempo de ejecución ampliada, el motor para base de datos dbExpress, un soporte excepcional de servicios Web y XML, un poderoso marco de trabajo de desarrollo Web, mas mejoras en el IDE y multitud de componentes y clases.

Delphi 7 proporcionó mas robustez a estas nuevas tecnologías con mejoras y arreglos (el soporte de SOAP y DataSnap) y ofrece soporte para tecnologías más novedosas (como los temas de Windows XP o UDDI), pero lo más importante es que permite disponer rápidamente de un interesante conjunto de herramientas de terceras partes: el motor de generación de informes RAVE, la tecnología de desarrollo de aplicaciones Web IntraWeb y el entorno de diseño ModelMaker. Finalmente, abre las puertas a un mundo nuevo al ofrecer (aunque sea como prueba) el primer compilador de Borland para el lenguaje Pascal Delphi no orientado a la CPU de Intel, si no a la plataforma CIL de .NET.

Delphi es una gran herramienta, pero es también un entorno de programación completo en el que hay muchos elementos involucrados.

3.4.4 Caracterización de las herramientas empleadas en el diseño

Para el diseño y modelación de procesos del Sistema APROV PLUS se usó el entorno de diseño ModelMaker 6, una herramienta de terceros incorporada por Borland a la versión 7 del entorno de desarrollo de Delphi.

ModelMaker representa una forma de nueva marca para desarrollar clases y paquetes de componentes para Borland Delphi. ModelMaker es una herramienta CASE estilo *UML* de productividad orientada a árboles de clases de doble dirección y refactorización, específicamente diseñada para generar código nativo para Delphi (de hecho éste fue hecho usando Delphi y ModelMaker). El lenguaje Object Pascal de Delphi está totalmente soportado por ModelMaker. Desde sus inicios fue diseñado para ser una herramienta inteligente y altamente productiva, que incorpora capacidades para realizar ingeniería inversa.

ModelMaker soporta la construcción de un grupo de diagramas UML y desde esta perspectiva se ve como una tradicional herramienta CASE. La clave de la magia de ModelMaker, su velocidad y poder, radica en su motor de modelado activo, el cual almacena y mantiene todas las relaciones entre las clases y sus miembros. Por ejemplo, renombrando una clase o cambiando su ancestro será inmediatamente propagado al código fuente generado automáticamente.

La principal diferencia entre ModelMaker y otra herramienta CASE es que el diseño está estrictamente ligado al código nativo expresado en Delphi. La principal diferencia entre ModelMaker y otros generadores de código para Delphi es su vista de alto nivel y las capacidades de reestructuración que le permiten conducir complejos diseños.

La única característica, actualmente no encontrada en ningún entorno de desarrollo y diseño para Delphi, es el soporte de patrones de diseño. Un buen número de patrones de diseño están implementados en ModelMaker como agentes activos "listos para usarse". Un "Patrón ModelMaker" no solamente insertará fragmentos de código estilo Delphi para implementar un patrón específico, sino que se mantendrá "con vida" para actualizar el código y reflejar cualquier cambio hecho en el diseño.

Como resultado, ModelMaker le permite:

- ✓ Desarrollo rápido
- ✓ Producción de diseños y códigos de inigualable calidad
- ✓ Diseño sin compromiso y refinamiento y experimentación con diseños hasta lograr lo deseado.
- ✓ Crear y mantener grandes módulos en poco tiempo
- ✓ Documentar sus diseños en diagramas estilo UML
- ✓ Documentar sus Units en archivos de ayuda con un simple clic.

Actualmente Modelmaker soporta los siguientes tipos de diagramas:

- ✓ Diagrama de clases: muestra la estructura estática del modelo, en particular los elementos que existen (como las clases e interfases), su estructura interna y sus relaciones con otros elementos. Los diagramas de clases no muestran información temporal, aunque pueden contener elementos que tiene o describen una apariencia

temporal. En ModelMaker, los diagramas de clases muestran relaciones entre clases, interfases y paquetes de componentes (herencia, composición, referencias, etc.).

- ✓ Diagramas de secuencia: muestran la dinámica de las interacciones entre objetos en diferentes roles. Un rol puede ser una clase, un actor, o puede venir desde fuera del sistema. El diagrama de secuencia incluye las secuencias cronológicas, pero no incluye las relaciones entre los objetos. Los diagramas de secuencia y los diagramas de colaboración expresan información similar, pero la muestran en formas diferentes. Los diagramas de secuencia muestran la secuencia explícita de los mensajes y resultan ser mejores cuando resulta importante visualizar el orden de los mensajes en el tiempo.
- ✓ Diagramas de colaboración: representan interacciones entre objetos como una serie de mensajes secuenciados. Los diagramas de colaboración describen tanto las estructuras estáticas como el comportamiento dinámico de un sistema.
- ✓ Diagramas de casos de uso: modelan la funcionalidad del sistema usando actores y casos de uso. El caso de uso representa la funcionalidad de un sistema, subsistema o clases, como manifestación de interactores externos con el sistema. Un diagrama de casos de uso es un grafo de actores, conjuntos de casos de usos y las relaciones entre estos elementos.
- ✓ Diagramas de estado: describen el comportamiento dinámico de entidades. Típicamente son usados para describir el comportamiento de las clases, auque pueden describir además el comportamiento de otras entidades del modelo como casos de uso, actores, subsistemas, operaciones o métodos.
- ✓ Diagrama de actividades: es una variación del diagrama de estado en el que el estado representa el comportamiento de acciones o subactividades. Ilustra la naturaleza dinámica del sistema modelando el flujo de control de inactividad en actividad. Una actividad representa una operación en alguna clase del sistema que resulta en un cambio en el estado del sistema. Típicamente un diagrama de actividad es usado para modelar el flujo de trabajo o proceso de negocio.
- ✓ Diagrama de paquetes: muestra las relaciones entre las "*Units*" de código Object Pascal implicadas en las cláusulas "*uses*". Este tipo de diagramas no están especificados en UML pero ModelMaker sí los implementa.
- ✓ Diagrama de implementación: pueden ser de dos tipos: de componentes y de despliegue. Los diagramas de componentes muestran las dependencias entre los componentes físicos del software incluyendo los clasificadores que lo especifican

(por ejemplo: implementación de clases) y los artefactos que lo implementan; como ficheros de código fuente, ficheros de código binario, ficheros ejecutables, scripts. Por su parte los diagramas de despliegue muestran la configuración del procesamiento de elementos en tiempo de corrida. Las instancias de los componentes de software representan manifestaciones en tiempo de corrida de unidades de código del software.

- ✓ Diagrama de mapas de ideas: consiste en una palabra o concepto central (nodo largo), alrededor del cual se dibujan varias ideas principales (nodos pequeños) relacionadas con este concepto. Se puede tomar cada idea relacionada y de nuevo dibujar varias ideas relacionadas a esta.

3.5 Descripción del proceso de ingeniería del sistema

3.5.1 Descripción del negocio

Los Sistemas de Cosecha de Madera incluyen dos actividades que son el aprovechamiento forestal y la construcción de caminos y acopiaderos. Cada actividad está constituida por una secuencia de etapas, cada una de ellas con una operación asociada. Estas operaciones se realizan usando tipos específicos de máquinas o herramientas. Las máquinas o herramientas se clasifican en cuatro categorías: Tractores, Camiones, Herramientas y Animales.

La Tabla 3.1 muestra que tipo de máquinas se pueden utilizar en cada una de las operaciones.

Tabla 3. 1. Uso de máquinas por operación

Operaciones	Tractores	Camiones	Herramientas	Animales
Actividad de Aprovechamiento				
Tala	x		x	
Tala-desrame	x		x	
Tala-desrame-troceado	x		x	
Desrame	x		x	
Desrame-troceado	x		x	
Troceado	x		x	
Extracción	x			x
Carga	x	x		x
Transporte	x	x		x
Carga-transporte	x	x		x

Actividad de Construcción de caminos y acopiaderos				
Levantamiento	x	x	x	x
Limpieza	x	x	x	x
Terraplén	x	x	x	x
Formación	x	x	x	x
Drenaje	x	x	x	x
Superficie	x	x	x	x
Acopiadero	x	x	x	x

* x: Se usa.

La actividad inicial tiene que ver con la construcción de la red de caminos y acopiaderos que se usarán como vías para extraer y transportar la madera. Luego le sigue la actividad de aprovechamiento cuya secuencia de etapas comienza por la tala y termina con el transporte.

Entre estas dos etapas se pueden definir diferentes combinaciones de otras etapas de acuerdo a varios flujos lógicos. Los flujos lógicos entre etapas se pueden representar mediante un grafo como lo muestra la Figura 3.6.

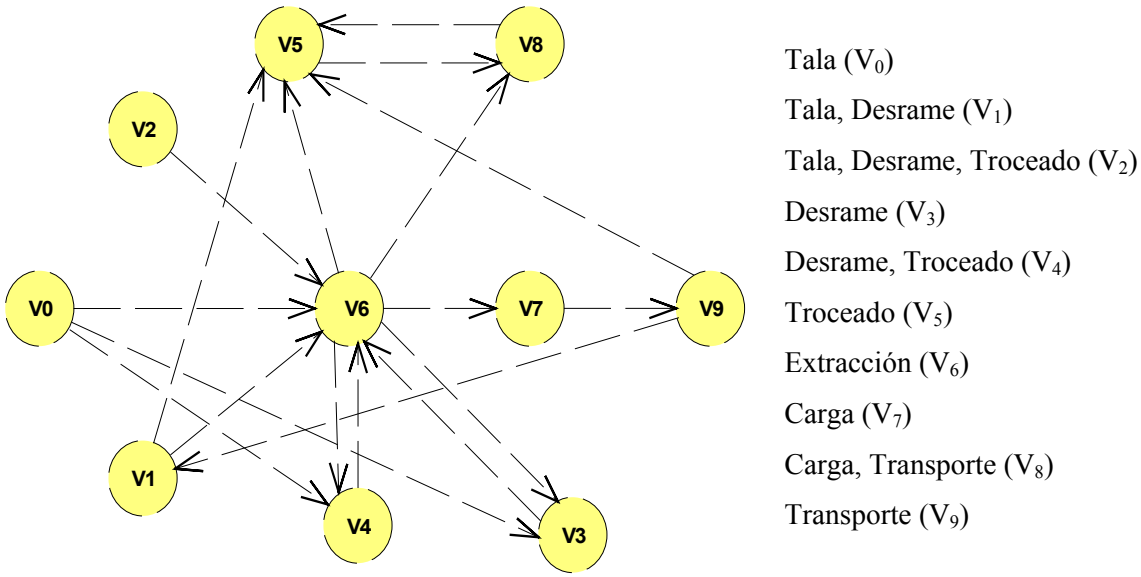
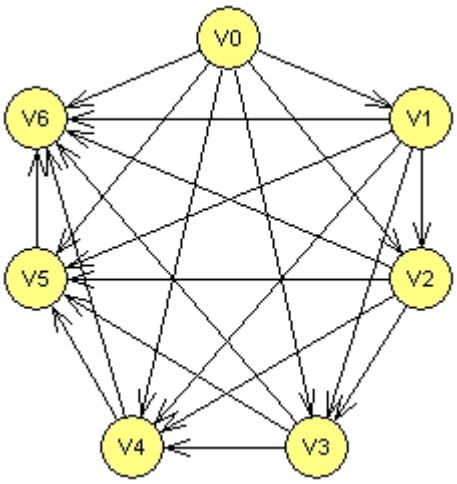


Figura 3. 7. Grafo dirigido que representa las combinaciones lógicas de las operaciones de aprovechamiento

La secuencia de operaciones para la actividad de construcción de caminos y acopiaderos se muestra en la Figura 3.7



Levantamiento (V_0)
Limpieza (V_1)
Terraplén (V_2)
Formación (V_3)
Drenaje (V_4)
Superficie (V_5)
Acopiadero (V_6)

Figura 3. 8. Grafo dirigido que representa las combinaciones lógicas de las operaciones de construcción de caminos y acopiaderos

Para las etapas se calcula un rendimiento en metros cúbicos por unidad de tiempo. Las fórmulas de rendimiento contienen varios parámetros cuyos valores deben de ser entrados. Cada etapa tiene una fórmula diferente.

Las máquinas o herramientas asociadas a las etapas tienen un costo de explotación que se expresa en pesos por unidad de tiempo. Las fórmulas para calcularla incluyen varios parámetros cuyos valores también tienen que ser entrados. Estos costos se dividen en tres categorías: costos fijos, costos de operación y costos de labor. El costo de explotación de una máquina es la suma de estos tres.

El costo de la etapa se calcula dividiendo el costo de explotación de la máquina entre el rendimiento de la etapa y se expresa en pesos por metro cúbico de madera. El costo de la actividad es la suma de los costos de las etapas, y el costo de la tecnología es la suma de los costos de sus dos actividades.

Las fórmulas para calcular el rendimiento de las etapas dependen de la metodología creada a partir de los estudios hechos para bosques que tienen determinadas características. De manera que si cambian estas características también pueden cambiar estas formulas.

El costo de la etapa de extracción en la actividad de aprovechamiento depende de las distancias medias entre caminos (S) y las distancias medias entre acopiaderos (L). La

actividad de construcción de caminos y acopiaderos completa depende también de ellos dos. Los incrementos o disminuciones de estas distancias medias producen efectos inversos en los costos de las dos actividades, por lo tanto no se puede establecer a priori que el costo total de la tecnología disminuye o aumenta al disminuir o aumentar las distancias medias. Por esa razón es necesario utilizar la modelación matemática para encontrar los valores de S y L que garantizan el costo mínimo de la tecnología, es decir la optimización del proceso desde el punto de vista de su eficiencia económica.

3.5.2 Diseño de los datos

Definición de Entidades y Atributos

Tecnología *(Descripción, Mínimo de distancia entre acopiaderos, Máximo de distancia entre acopiaderos, Mínimo de distancia entre caminos, Máximo de distancia entre caminos, Volumen de madera):* almacena las tecnologías de aprovechamiento que se crean con el sistema APROV PLUS.

Secuencia *(Tipo):* almacena las secuencias de operaciones de las tecnologías que se crean.

Etapa *(Operación, Equipo, Formula de rendimiento):* almacena las etapas de cada secuencia de operaciones.

Parámetros de rendimiento *(Nombre, Valor, Descripción):* almacena los parámetros necesarios para calcular el rendimiento de cada etapa.

Tractor *(Nombre, Parámetros para cálculo de rendimiento):* agrupa la lista de tractores usados para la cosecha de madera.

Herramienta *(Nombre, Parámetros para cálculo de rendimiento):* agrupa la lista de herramientas usadas para la cosecha de madera.

Camión *(Nombre, Parámetros para cálculo de rendimiento):* agrupa la lista de camiones usados para la cosecha de madera.

Animal *(Nombre, Parámetros para cálculo de rendimiento)*: agrupa la lista de animales usados para la cosecha de madera.

Por la abstracción de características (atributos comunes) de estas 4 entidades, se acude a la extensión del MER "generalización / especialización" quedando como entidad generalizadora "Equipos" y el resto como especializaciones de la forma siguiente:

Equipos *(Nombre)*

> ***Tractor*** *(Parámetros para cálculo de rendimiento)*
>
> ***Camión*** *(Parámetros para cálculo de rendimiento)*
>
> ***Animal*** *(Parámetros para cálculo de rendimiento)*
>
> ***Herramienta*** *(Parámetros para cálculo de rendimiento)*

Del análisis de los roles de estas entidades obtenemos su clasificación en las categorías siguientes:

Entidad	**Categoría**
Tecnología.....	Indicador
Secuencia.....	Indicador
Etapa.....	Indicador
Parámetros de rendimiento.	Indicador
Equipos.....	Codificador
Tractor.....	Codificador
Camión.....	Codificador
Animal.....	Codificador
Herramienta.....	Codificador

Definición de relaciones y sus atributos

R1 | *Tecnología & Secuencia*

Es la relación 1 a n entre tecnología y secuencia. Una tecnología esta formada por n secuencia de operaciones.

R2 | *Secuencia & Etapa*

Es la relación 1 a n entre secuencia y etapa. Una secuencia de operaciones consta de varias etapas.

R3 | *Etapa & Parámetros de rendimiento*

Es la relación 1 a n entre etapa y parámetros del rendimiento. El rendimiento de una etapa está determinado por varios parámetros.

R4 | *Equipos & Etapa*

Es la relación 1 a n entre equipos y etapas. Un equipo puede ser usado en varias etapas de una tecnología.

La forma más sintética de representar los elementos derivados del modelo conceptual es a través de un diagrama. Existen diferentes vías para diagramar el modelo conceptual. Las más utilizadas son los diagramas de Chen y los diagramas UML. El presente trabajo recurre a los diagramas de Chen porque son simples y efectivos y son ampliamente utilizados para comunicar diseños de datos. A partir de los elementos obtenidos en el diseño conceptual, se propone el Diagrama Entidad-Relación (DER) que aparece en la figura 3.8.

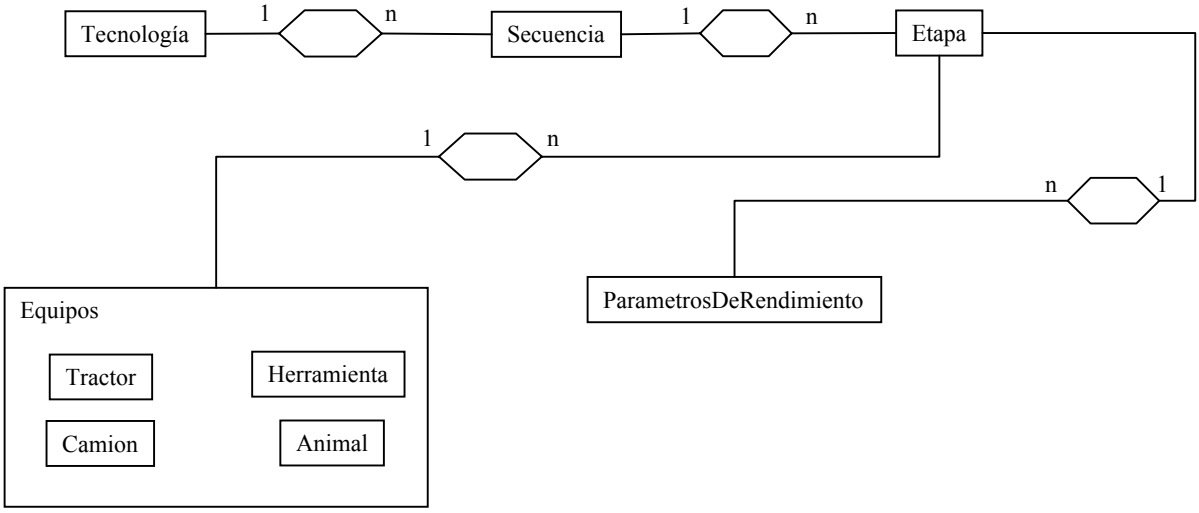


Figura 3. 9. Diagrama Entidad Relación

Para acometer el Modelo Lógico Global de los Datos, están disponibles varias herramientas. En el presente trabajo se escogieron dos de ellas, la definición de relaciones de atributos que derivan en tablas físicas (usando el lenguaje de implementación de bases de datos relacionales) y las tarjetas de objetos (Apéndice A).

Tablas y relaciones que se derivan del DER

Tecnología (ID, Descripción, SL, SH, LL, LH, VolMad).
Secuencia (ID, Tipo, Tecnologia)
Etapa (ID, Operación, Secuencia, Equipo, FormulaDeRendimiento, Alquilado)
Parámetros de rendimiento (ID, Nombre, Valor, Descripción, Etapa)
Equipos (ID, Descripcion, Tipo)

Tractor (ID, FijPrecComp, FijValResid, FijVidaUtil, FijDiaTrabAn, FijHorTrabDia, FijTasaInter, FijTasaImpAn, OperPcDepHerrMantRep, OperConsCombHor, OperValLitComb, OperPcConsCombEstConsLub, OperValLitLub, OperValOtMat, OperVidUtOtMatHor, OperValCab, OperVidUtCabHor, OperValWin, OperVidUtWinHor, OperValOrugNeum, OperVidUtOrugNeumHor, LabSalBasOper, LabSalBasAyud, LabCantAyud, LabBenAdic, LabHorNoTrab, LabHorTrab, LabPCientCosDir, AlqCostFijo, AlqCostOper, AlqCostLabor)

Herramienta (ID, FijPrecComp, FijValResid, FijVidaUtil, FijDiaTrabAn, FijHorTrabDia, FijTasaInter, FijTasaImpAn, OperPcDepHerrMantRep, OperConsCombHor, OperValLitComb, OperPcConsCombEstConsLub, OperValLitLub, OperValOtMat, OperVidUtOtMatHor, OperValSab, OperVidUtSabHor, OperValCad, OperVidUtCadHor, LabSalBasOper, LabSalBasAyud, LabCantAyud, LabBenAdic, LabHorNoTrab, LabHorTrab, LabPCientCosDir, AlqCostFijo, AlqCostOper, AlqCostLabor)

Camión (ID, FijPrecComp, FijValResid, FijVidaUtil, FijDiaTrabAn, FijHorTrabDia, FijTasaInter, FijTasaImpAn, OperPcDepHerrMantRep, OperConsCombHor, OperValLitComb, OperPcConsCombEstConsLub, OperValLitLub, OperValOtMat, OperVidUtOtMatHor, OperValNeum, OperCantNeum, OperVidUtNeum, LabSalBasOper, LabSalBasAyud, LabCantAyud, LabBenAdic, LabHorNoTrab, LabHorTrab, LabPCientCosDir, AlqCostFijo, AlqCostOper, AlqCostLabor)

Animal (ID, FijPrecComp, FijValResid, FijVidaUtil, FijDiaTrabAn, FijHorTrabDia, FijTasaInter, FijTasaImpAn, FijValAlimAnim, FijValMedVeter, OperValCompArn, OperValResArn, OperVidUtArn, OperValCompMis, OperValResMis, OperVidUtMisAn, OperValAlimSupAn, OperValMedSupAn, LabSalBasOper, LabSalBasAyud, LabCantAyud, LabBenAdic, LabHorNoTrab, LabHorTrab, LabPCientCosDir, AlqCostFijo, AlqCostOper, AlqCostLabor)

3.5.3 Ingeniería del Sistema

Requerimientos funcionales del sistema

R1 Autentificación de usuarios

R2 Gestión de la cuenta de usuario

R3 Gestión de registro de equipos

R4 Crear tecnología de aprovechamiento forestal

R5 Insertar operaciones

R6 Insertar equipos

R7 Abrir tecnología existente

R8 Optimizar costos de la tecnología

R9 Obtener cuadro resumen de costos y gráficos

Actores (beneficiarios) de APROV PLUS

Tabla 3. 2. Descripción del rol de los beneficiarios.

Actor	Rol
Usuario	Único actor y responsable de la actualización del Registro de Equipos y sus costos, así como de la creación y optimización de las tecnologías. Puede crear, almacenar y recuperar tecnologías de aprovechamiento forestal en y desde la base de datos. Puede cambiar la contraseña de acceso al sistema.

Paquetes de Casos de Uso

Los Casos de Uso del sistema propuestos (fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para los actores) han sido agrupados en paquetes de Casos de Uso siguiendo el criterio de funcionalidad con el objetivo de lograr una mejor comprensión del Modelo y modularización de las funcionalidades que brinda el sistema.

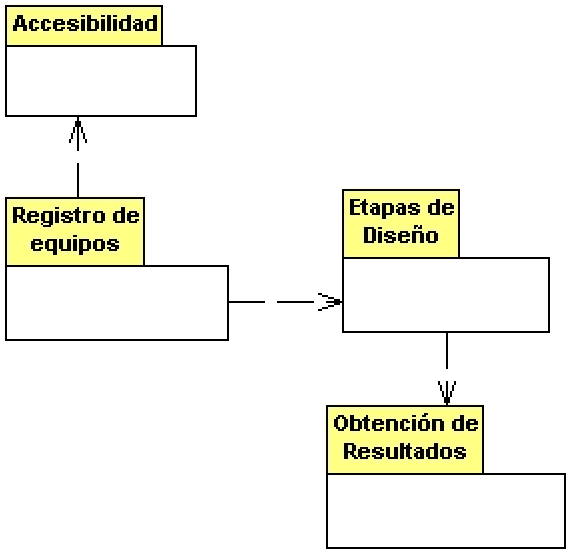


Figura 3. 10. Diagrama de paquetes de Casos de Uso de APROV PLUS

Se han definido 4 paquetes que garantizan un análisis modular del sistema. Los paquetes son Accesibilidad, Registro de equipos, Etapas de Diseño y Obtención de Resultados. Como se muestra cada uno de los paquetes están interconectados entre sí representando la comunicación que existe entre ellos.

El paquete **Accesibilidad** contiene los Casos de Uso: Autentificar Usuario y Cambiar Contraseña de Acceso.

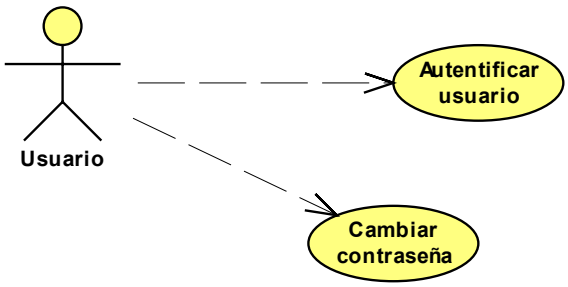


Figura 3. 11. Paquete "Accesibilidad" de APROV PLUS

Tabla 3. 3. Descripción del caso de uso Autentificar usuario.

Caso de uso:	**Autentificar Usuario.**
Actores:	Usuario (inicia)
Descripción: El Caso de Uso se inicia cuando el usuario necesita hacer uso del sistema APROV PLUS, una vez realizada su autentificación concluye el Caso de Uso.	
Referencias:	R1
Precondiciones:	

Poscondiciones:	Se inicia o no la sesión del usuario en correspondencia si la autentificación es correcta.
Curso normal de los eventos	
Acción del Usuario	**Respuesta de APROV PLUS**
1 Necesita hacer uso de APROV PLUS	2 Presenta la interfase Autenticación A (Apéndice B)
3 Introduce su Cuenta y contraseña y autentifica ejecutando el botón "Aceptar"	4 Si la autentificación es correcta se finaliza el Caso de Uso cerrando la interfase de autentificación y mostrando la interfase principal de APROV PLUS IP (Apéndice B)
5 Selecciona el botón cancelar de la interfase de Autentificación.	Cierra la interfase de Autentificación y finaliza el Caso de Uso

Tabla 3. 4. Descripción del caso de uso Actualizar cuenta de usuario.

Caso de uso:	**Actualizar Cuenta de Usuario.**
Actores:	Usuario (Inicia).
Descripción: El Caso de Uso es iniciado por el Usuario cuando desea cambiar la contraseña de entrada al sistema APROV PLUS. Realizada la acción finaliza el Caso de Uso.	
Referencias:	R2
Precondiciones:	El usuario autenticado debe tener derechos de acceso.
Poscondiciones:	Se actualiza Registro de Cuentas de Usuarios
Curso normal de los eventos	
Acción del Usuario	Respuesta del APROV PLUS
1 Selecciona la opción Cambiar contraseña del Menú Herramientas de la interfase principal de APROV PLUS	2 Presenta la interfase Cambiar contraseña C (Apéndice B)
3 Escribe contraseña anterior, nueva contraseña y confirmación de nueva contraseña en C.	4 Cambia la contraseña de acceso al sistema.

El paquete **Registro de Equipos** contiene el Caso de Uso Gestionar Registro de Equipos el cual posee como Casos de Usos extendidos: Adicionar, Modificar y Eliminar. En este paquete se trata la gestión de los parámetros necesarios en cada equipo para calcular sus costos de explotación y de rendimiento.

Caso de uso Registro de equipos

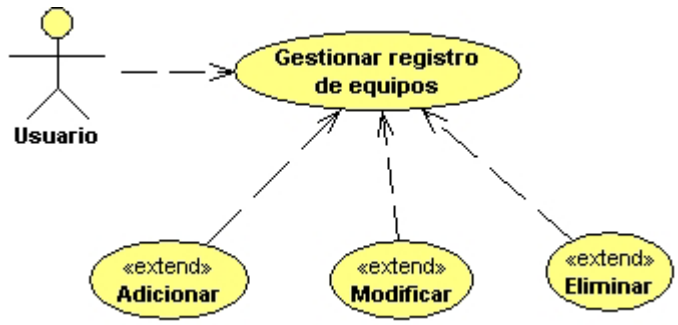


Figura 3. 12. Caso de uso "Registro de equipos" de APROV PLUS

Tabla 3. 5. Descripción del caso Gestionar Registro de Equipos.

Caso de uso:	**Gestionar Registro de Equipos.**		
Actores:	Usuario (inicia)		
Descripción: El Caso de Uso es iniciado por el Usuario cuando, al intentar crear nuevas tecnologías, detecta que no existen equipos para asociar a las operaciones que conforman las secuencias. En ese caso se procede a actualizar el Registro de Equipos (ya sean Tractores, Camiones, Herramientas o Animal) Realizada la acción finaliza el Caso de Uso.			
Referencias:	R3		
Precondiciones:			
Poscondiciones:	Se actualiza el Registro de Equipos		
Curso normal de los eventos			
Acción del Usuario		**Respuesta de APROV PLUS**	
1	Selecciona Lista de Equipos del menú Ver de la interfase principal	2	Presenta la interfase E (Apéndice B)
3	El usuario selecciona el Tipo de equipo.	4	El sistema muestra los equipos disponibles para el tipo seleccionado.
5	El usuario selecciona el equipo que desea actualizar	6	El sistema muestra los parámetros para calcular los costos.
7	El usuario puede cambiar o agregar valores a los parámetros de los costos con y actualizarlos	6	El sistema actualiza los parámetros y calcula los costos fijos, de operación y labor del equipo seleccionado.
Sección Nuevo			
1	El usuario necesita agregar equipo	2	El sistema crea un registro en blanco para almacenar características del nuevo equipo y pregunta el nombre del equipo
3	El usuario especifica los parámetros que se necesitan	4	El sistema guarda el registro del nuevo equipo.

	para calcular los costos		
Sección Eliminar			
1	El usuario necesita eliminar un equipo existente	2	El sistema solicita la confirmación de eliminación del registro correspondiente al equipo activo.
3	El usuario confirma la eliminación del equipo o decide que no va a eliminarlo.		**Si se confirma eliminación**, el Sistema verifica si el equipo está asociado a alguna operación dentro de las tecnologías existentes, de ser así se eliminarán los datos del equipo y de las operaciones relacionadas con este.

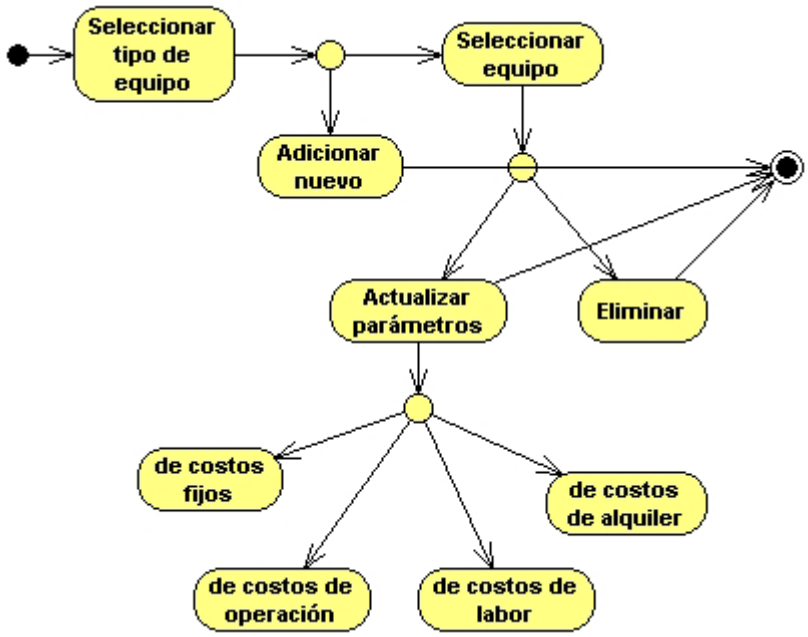


Figura 3. 13. Diagrama de estado para el caso de uso Registro de equipos

El paquete **Diseño de tecnología** incluye los Casos de Uso:

Crear Tecnología que posee a su vez como casos Especializados a:

Actividades de construcción de caminos y acopiaderos

Actividades de aprovechamiento forestal

El caso de uso Fórmulas y parámetros del rendimiento

El caso de uso Insertar Operación

El caso de uso Insertar Equipos

El caso de uso Abrir tecnología existente

El caso de uso Optimizar costos de la tecnología

Caso de uso "Crear Tecnología"

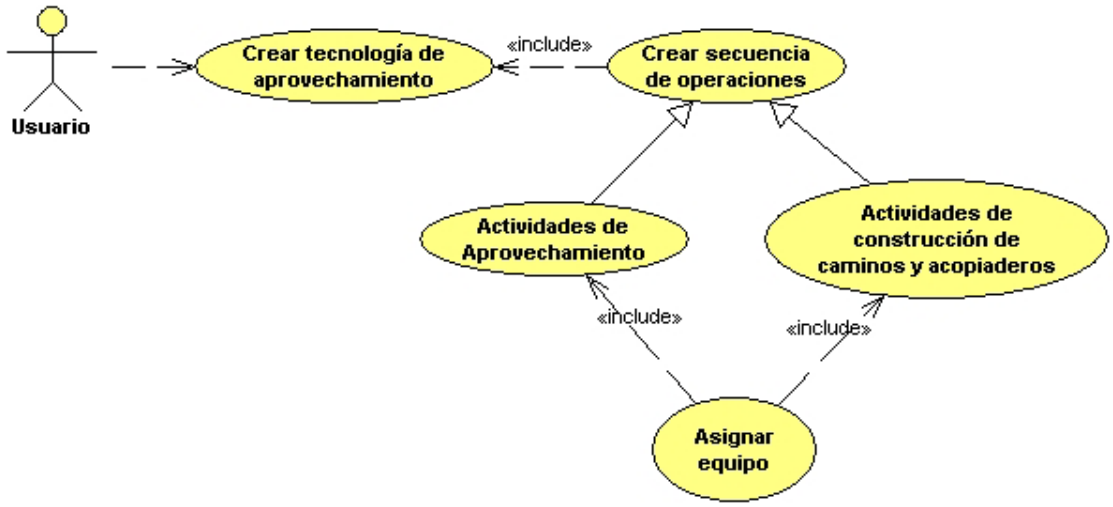


Figura 3. 14. Caso de uso "Crear Secuencia de Operaciones" de APROV PLUS

Tabla 3. 6. Descripción del caso Crear tecnología

Caso de uso:	**Crear Tecnología**
Actores:	Usuario (inicia)
Descripción: El Caso de Uso es iniciado por el Usuario cuando necesita crear una nueva tecnología de aprovechamiento. Realizada la acción finaliza el Caso de Uso.	
Referencias:	R4
Precondiciones:	Tener equipos asociados a las distintas operaciones que conformarán la tecnología de aprovechamiento.
Poscondiciones:	Se crea una nueva tecnología
Curso normal de los eventos	
Acción del Usuario	**Respuesta de APROV PLUS**
1 Selecciona Nueva Tecnología del menú Archivo de la interfase principal	2 Presenta la interfase L (Apéndice B)
3 El usuario selecciona las operaciones de las actividades a incluir en la secuencia de operaciones que forman la tecnología	4 Se activa el botón "Insertar".
5 El usuario necesita insertar la operación seleccionada.	6 El sistema: Inserta la operación seleccionada en la secuencia de operaciones y muestra su representación gráfica en el contenedor de secuencias de la interfase principal. Muestra la interfase AE (Apéndice B) para asignar equipos.

Sección Asignar equipo			
7	El usuario puede seleccionar el equipo a asignar.	8	El sistema carga en memoria los datos del equipo seleccionado.
9	El usuario puede: Asignar un equipo a la operación y cerrar la ventana. Cerrar la interfase sin agregar equipo.	10	En caso de: Asignar equipo: el sistema agrega el equipo a la operación seleccionada en el punto 3 y regresa al punto 2 si quedan operaciones que asignar. Cerrar: el sistema regresa al punto 2 si quedan operaciones que asignar.

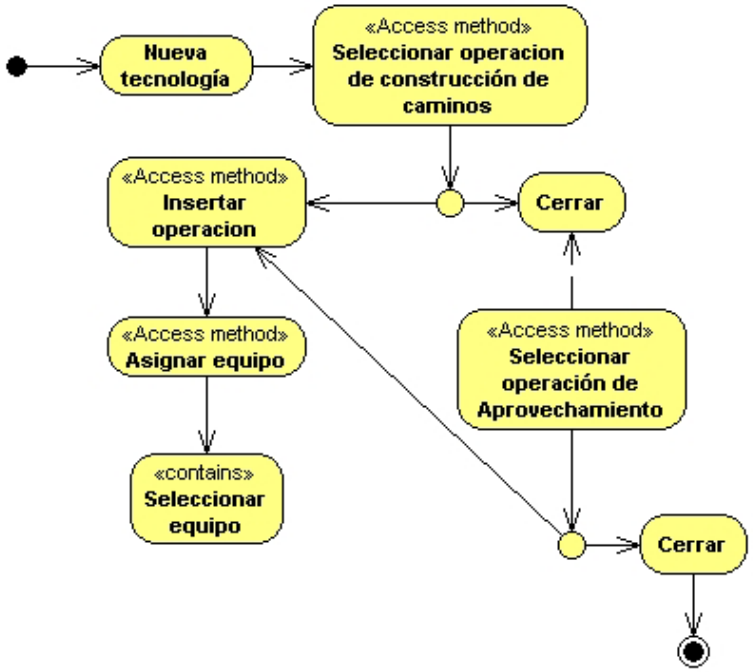


Figura 3. 15. Diagrama de estado para el caso de uso Crear tecnología

Caso de uso Fórmulas y parámetros del rendimiento

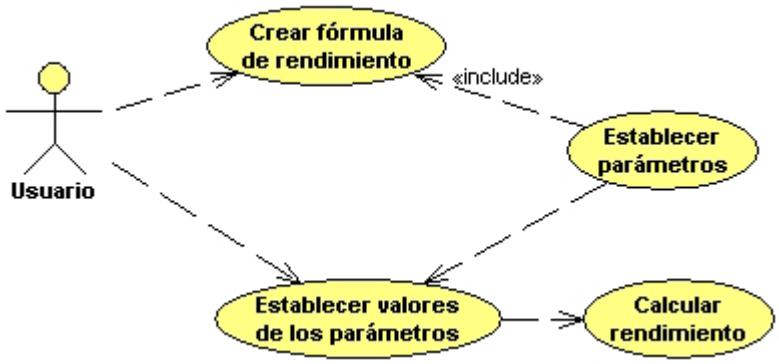


Figura 3. 16. Caso de uso "Crear fórmulas y parámetros de rendimiento

Tabla 3. 7. Descripción del caso de uso Fórmulas y parámetros del rendimiento

Caso de uso:	**Fórmulas y parámetros del rendimiento**
Actores:	Usuario (inicia)
Descripción: El Caso de Uso es iniciado por el Usuario cuando necesita asignar una fórmula al cálculo del rendimiento de una etapa y asignar valores a sus parámetros. Realizada la acción finaliza el Caso de Uso.	
Referencias:	R4
Precondiciones:	Tener al menos una etapa insertada en al menos una actividad.
Poscondiciones:	Se establecen los valores de los parámetros del rendimiento para la fórmula por defecto o se crea una nueva fórmula de rendimiento.
Curso normal de los eventos	
11 El usuario necesita: Asignar valores a los parámetros de las fórmulas de rendimiento establecidas por defecto. Crear nueva fórmula de rendimiento, a partir del menú contextual de una etapa.	12 El sistema muestra la interfase P (Apéndice B)
13 Al mostrar la interfase P el usuario puede: Entrar los valores de los parámetros de rendimiento de la fórmula por defecto. Crear una nueva fórmula y entrar los valores de los parámetros definidos.	14 El sistema asigna la fórmula al rendimiento de la etapa seleccionada y lo calcula.

Caso de uso Insertar operación

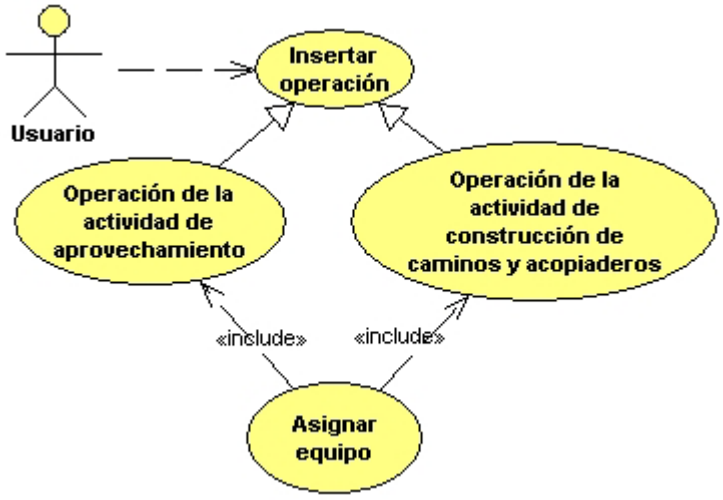


Figura 3. 17, Caso de uso "Insertar operación" de APROV PLUS

Tabla 3. 8. Descripción del caso de uso Insertar operación

Caso de uso:	**Insertar operación.**
Actores:	Usuario (inicia)
Descripción: El Caso de Uso es iniciado por el Usuario cuando necesita insertar una operación a la secuencia activa de la tecnología en uso. Realizada la acción finaliza el Caso de Uso.	
Referencias:	R5
Precondiciones:	Tener creada una tecnología
Poscondiciones:	Si la secuencia activa tiene incorporada al menos una operación, se inserta una a continuación de la existente, de lo contrario se inserta una como la primera operación de la secuencia.
Curso normal de los eventos	
Acción del Usuario	**Respuesta de APROV PLUS**
1 El usuario selecciona Operación del menú Insertar de la interfase principal IP	2 Presenta la interfase L (Apéndice B) Se continúa en el punto 3 del Caso de Uso Crear tecnología de aprovechamiento (R4)

Caso de uso Insertar equipo

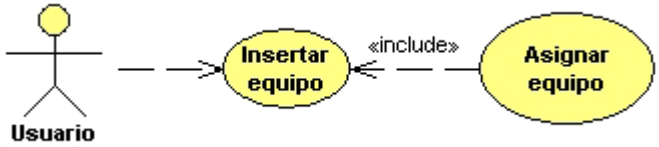


Figura 3. 18. Caso de uso "Insertar Equipo" de APROV PLUS

Tabla 3. 9. Descripción del caso de uso Insertar equipo

Caso de uso:	**Insertar Equipos.**
Actores:	Usuario (inicia)
Descripción: El Caso de Uso es iniciado por el Usuario cuando necesita insertar un equipo a una operación de la secuencia activa de la tecnología en uso. Realizada la acción finaliza el Caso de Uso.	
Referencias:	R6
Precondiciones:	Tener seleccionada una operación
Poscondiciones:	Se inserta un equipo a la operación seleccionada.
Curso normal de los eventos	
Acción del Usuario	**Respuesta de APROV PLUS**
1 El usuario selecciona Equipos del menú Insertar de la interfase principal IP	2 Presenta la interfase AE (Apéndice B) Se continúa en el punto 7 del Caso de Uso Crear tecnología de aprovechamiento (R4)

Caso de uso Abrir Tecnología

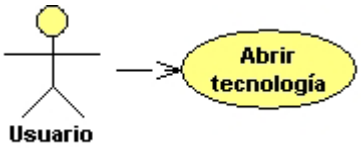


Figura 3. 19. Caso de uso "Abrir Tecnología existente" de APROV PLUS

Tabla 3. 10. Descripción del caso de uso Abrir tecnología existente

Caso de uso:	**Abrir tecnología existente**
Actores:	Usuario (inicia)
Descripción: El Caso de Uso es iniciado por el Usuario cuando necesita abrir una tecnología previamente creada. Realizada la acción finaliza el Caso de Uso.	
Referencias:	R7
Precondiciones:	Que exista al menos una tecnología almacenada en la base de datos.
Poscondiciones:	Se carga la tecnología en el contenedor.
Curso normal de los eventos	
Acción del Usuario	**Respuesta de APROV PLUS**
1 El usuario selecciona Abrir tecnología del menú Archivo de la interfase principal.	2 Presenta la interfase T (Apéndice B) donde se muestra la lista de tecnologías almacenadas en la base de datos.
3 El usuario puede seleccionar la tecnología a cargar.	4 El sistema activa los botones "Aceptar" y "Eliminar" para cargar o eliminar la tecnología seleccionada.

5	El usuario puede: Cargar en el contenedor de la interfase principal la tecnología seleccionada. Eliminar de la base de datos la tecnología seleccionada. Cerrar la interfase T sin cargar tecnología.	6	En caso de: Aceptar: el sistema carga y muestra la tecnología en el contenedor de la interfase principal. Eliminar: el sistema elimina de la base de datos la tecnología seleccionada. Cerrar: el sistema cierra la interfase sin cargar la tecnología.

El paquete "**Obtención de Resultados**" contiene los casos de uso "Optimizar tecnología" y "Obtener cuadro resumen y gráficos".

Caso de uso Optimizar tecnología

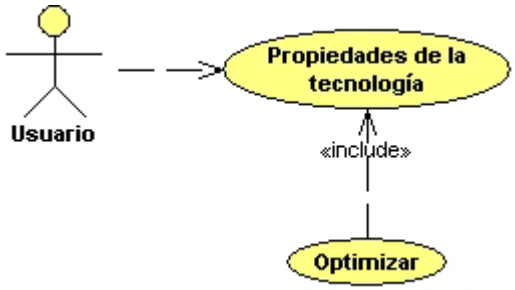


Figura 3. 20. Caso de uso Optimizar costos de la tecnología de APROV PLUS

Tabla 3. 11. Descripción del caso de uso optimizar costos de la tecnología

Caso de uso:	**Optimizar costos de la tecnología.**		
Actores:	Usuario (inicia)		
Descripción: El Caso de Uso es iniciado por el Usuario cuando necesita optimizar la tecnología activa. Realizada la acción finaliza el Caso de Uso.			
Referencias:	R8		
Precondiciones:	Que exista una tecnología cargada en el contenedor de la interfase principal.		
Poscondiciones:	Se optimiza la tecnología sobre la base costos mínimos.		
Curso normal de los eventos			
Acción del Usuario		**Respuesta de APROV PLUS**	
1	El usuario selecciona Propiedades de la tecnología del menú Archivo de la interfase principal.	2	Presenta la interfase O (Apéndice B) donde se muestran las propiedades de la tecnología activa.
3	El usuario necesita entrar el volumen de madera a aprovechar.	4	El sistema fija el volumen de madera entrado para calcular los costos de la tecnología.
5	El usuario puede marcar la opción Optimizar costo.	6	El sistema activa los elementos que intervienen en la optimización.

7	El usuario puede establecer los rangos de distancias medias entre caminos y acopiaderos que influyen en los costos de la tecnología.	8	El sistema fija los valores entrados como los rangos de distancias medias entre caminos y acopiaderos.
9	El usuario puede optimizar los costos.	10	El sistema calcula los costos unitarios de las operaciones y de la tecnología y corre el modelo matemático que encuentra la distancia entre caminos y acopiaderos optima para costos mínimos. Muestra la interfase R (Apéndice B)
Curso alternativo de los eventos			
2	El usuario puede ver el cuadro resumen de costos de la tecnología.	3	El sistema muestra la interfase R calculando previamente los costos unitarios de las operaciones y la tecnología.
4	El usuario puede ver el gráfico de costos de la tecnología.	5	El sistema muestra la interfase G (Apéndice B)

Caso de uso Obtener cuadro resumen y gráficos

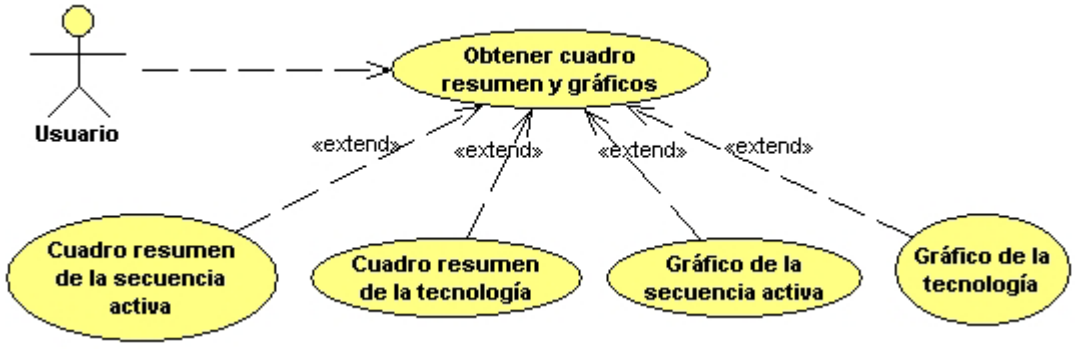


Figura 3. 21. Caso de uso "Obtener cuadro resumen y gráficos" de APROV PLUS

Tabla 3. 12. Descripción del caso de uso Obtener cuadros resumen y gráficos

Caso de uso:	**Obtener cuadros resumen y gráficos**
Actores:	Usuario (inicia)
Descripción: El Caso de Uso es iniciado por el Usuario cuando necesita obtener los resúmenes y gráficos que muestran los costos de las secuencias y la tecnología. Realizada la acción finaliza el Caso de Uso.	
Referencias:	R9
Precondiciones:	Que exista al menos una tecnología cargada en el contenedor de la interfase principal.
Poscondiciones:	Se muestran los cuadros resúmenes y gráficos que muestran los costos de las secuencias y la tecnología.

Curso normal de los eventos	
Acción del Usuario	**Respuesta de APROV PLUS**
1 El usuario puede: Seleccionar Gráfico de la secuencia activa del menú Ver de la interfase principal. Seleccionar Gráfico de la tecnología del menú Ver de la interfase principal. Seleccionar Cuadro resumen de la secuencia activa del menú Ver de la interfase principal. Seleccionar Cuadro resumen de la tecnología del menú Ver de la interfase principal.	2 El sistema calcula los costos unitarios de las operaciones y de la tecnología. En caso de: Seleccionar Gráfico de la secuencia activa: el sistema muestra la interfase G con los datos de la secuencia activa. Seleccionar Gráfico de la tecnología: el sistema muestra la interfase G con los datos de la tecnología. Seleccionar Cuadro resumen de la secuencia activa: el sistema muestra la interfase R con los datos de la secuencia activa. Seleccionar Cuadro resumen de la tecnología: el sistema muestra la interfase R con los datos de la tecnología.

Principales clases que intervienen en la creación de una tecnología de aprovechamiento forestal.

En la creación de tecnologías de aprovechamiento forestal interviene un conjunto de Clases que proporcionan los servicios necesarios para llevar a cabo el mencionado proceso.

Para cada uno de los elementos que conforman la tecnología se creó una clase, partiendo de la operación básica hasta la propia tecnología. En la figura 3.22 se representa el diagrama que contiene estas clases.

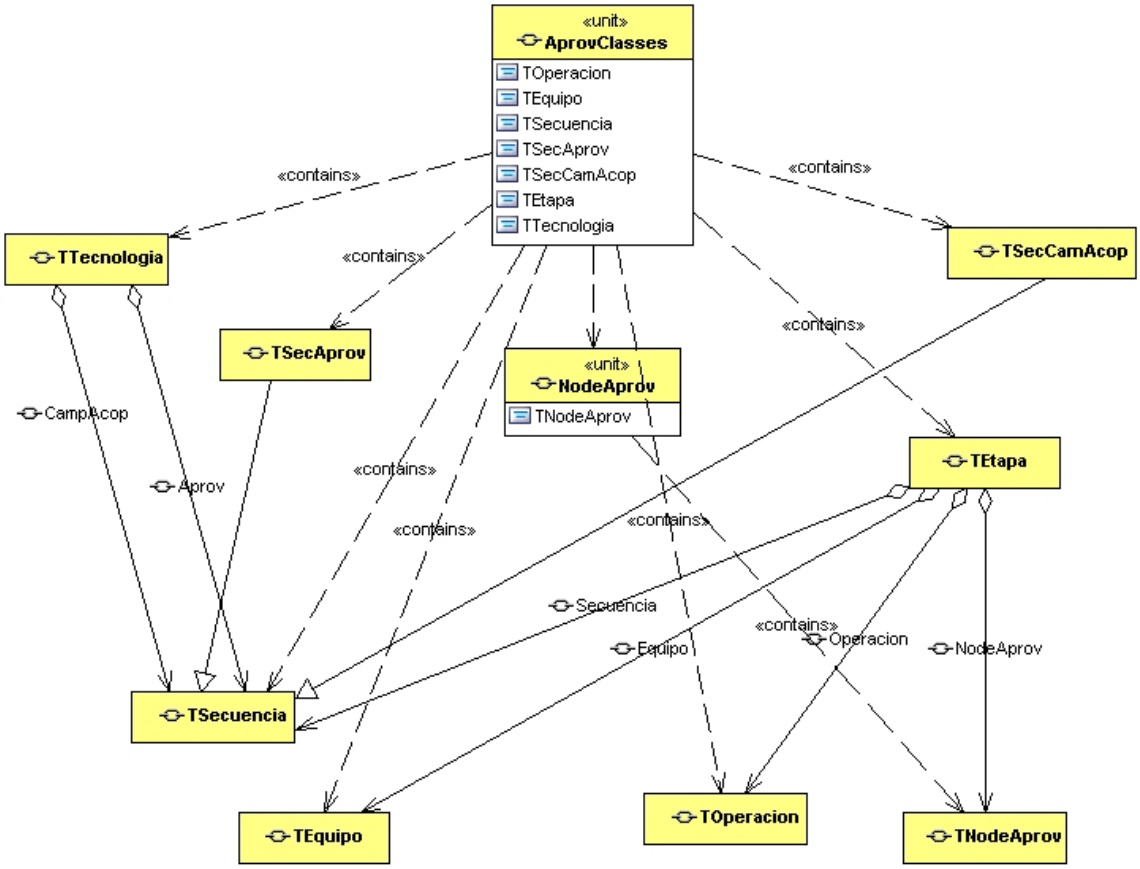


Figura 3. 22. Diagrama de clases de APROV PLUS

4. CONSIDERACIONES FINALES

Cumplimiento de La Relación: Objeto-Problema-Objetivos-Métodos-Resultados.

Se concluyó la construcción de un modelo matemático que permite optimizar y evaluar la eficiencia de los sistemas de cosecha de madera y se diseñó e implementó un software que permite automatizar el proceso de diseño de sistemas de cosecha de madera y resolver el modelo matemático. Estas herramientas pueden ser aplicadas en las clases de aprovechamiento forestal del tercer año de la carrera de ingeniería forestal, para desarrollar habilidades en los estudiantes en la creación de sistemas de cosechas y la toma de decisiones al respecto para alcanzar la eficiencia económica de estos procesos.

Aplicabilidad de la propuesta.

Los requerimientos del software hacen posible su instalación y uso en los laboratorios docentes de la facultad forestal y unidades docentes de la universidad de Pinar del Río y en cualquier PC sobre Windows con rendimiento promedio. Esto lo hace disponible para los estudiantes de la carrera que reciben la asignatura de Aprovechamiento Forestal.

Valoración acerca de la adquisición de conocimientos y desarrollo de habilidades por parte del maestrante.

- ✓ Se profundizó en el uso de ModelMaker como herramienta CASE en la obtención de los Diagramas de Casos de Uso.
- ✓ Se profundizó en el enfoque relacional para la modelación de bases de datos, así como en el manejo del SGBD MS Access.
- ✓ Se alcanzó un nivel aceptable de habilidades en el uso de técnicas de programación vinculadas al paradigma de la programación orientada a objetos, usando el entorno de programación Borland Delphi.

Limitaciones identificadas

- ✓ Profundizar en la confección de la Ayuda al Sistema presentada
- ✓ Proyectarse en su aplicabilidad en la producción

5. BIBLIOGRAFÍA

[1] Appel, K. y W. Hanken, W. (1977): "The solution of the four-color map problem", *Scientific American*, Illinois, Journal of Mathematics, pp. 108-121, 1977.

[2] Appel, K. y W. Hanken. (1977): "Every planar map is four colorable". Part I: *Discharging,* Illinois, Journal of Mathematics, pp. 429-490.

[3] Birthwistle, G. M., et-al, (1973): *SIMULA Begin*, Philadelphia, Auerbach Press.

[4] Cándano, F. (2004): *Aprovechamiento Forestal*. Cuba, Editorial Félix.

[5] Dijkstra, E. W. (1972): *Structured Programming*. Nueva York, Academic Press.

[6] Dikstra, D. P. and R. Heinrich (1988): Model Code of Forest Harvesting Practice, Rome, Italy, FAO.

[7] FAO. (1979): "Planificación de carreteras forestales y sistemas de aprovechamiento". Estudio FAO, *FAO,* 2, pp.177.

[8] Forte, S. and T. Howe (2001): Access 2002 Development, US, SAMS, pp. 15-147.

[9] Goguen, J. A., et-al. (1997): "An Initial algebra approach to the specifications, correctness and implementations of abstract data types". *Current Trends in Programming Methology*, volumen 4, Englewood Cliffs, NJ, Prentice Hall.

[10] Greene, W. D and J. F. McNeel. (1991): *Productivity and cost of sawhead fellerbunchers in the South Forestry Production*, Jun 41 (3), pp. 21-26.

[11] Guttag, J. (1997): "Abstract data types and the development of data structures". *Comunicacionales de la ACM*, pp. 396-404.

[12] Heileman, G. L. (1998): *Estructuras de datos, Algoritmos, y Programación Orientada a objetos*. España.

[13] Jacobson, I. et-al (2000): *El Proceso Unificado de Desarrollo de Software*, Madrid, Pearson Education.

[14] Kantorovich, L. V. (1999): *Las matemáticas en la economía: Logros, dificultades, perspectivas,* Nueva York.

[15] Lanford, B. L and B. J. Stokes. "Cost and productivity comparison of two thinning systems", *APA technology*, US, Palpwood Associations, pp. 2.

[16] Pérez, D. (2003): *Database design for forest fires localization and analysis*, The Nederlands, Intenational Institute for Geo-information Science and Earth Observations.

[17] Peterson, G. E. (1987): *Object – Oriented Computing*, volumen 1, Washington, DC, IEEE Computer Society Press.

[18] Sessions, J. and Y. Huat (1989): "Optimimizin road spacing and equipment allocation simultaneously", *Forest Products Journal*, Vol. 39, No 10, Oregon, Corvallis, pp. 43-46.

[19] Silva, C. E; C. Cardoso y A. Paulo (1991): "Optimización de la productividad y el costo de extracción forestal con winche arrastrador", *I Simposio Brasileño sobre explotación y transporte forestal*. Universidad Federal de Vicosa, Belo Horizonte, Minas Gerais, Brasil, pp.15.

[20] Stokes, B. J; W. F. Watson; A. A. (1989): "Twaddle and I. C. Ira Cameron, Production and cost for in woods flail processing of southern pines", *ASAE paper 89-7592. St. Joseph*, US, MI, American Society of Agricultural Engineers, pp.24.

[21] Stroustrup, B. (1986): *The C++ Programming Language*. MA, Addison-Wesley, Reading.

APÉNDICE A

Tarjetas de Objetos

Objeto "***Tecnología***"

Atributo	Tipo	Restricción	Descripción
ID	Autonumérico	Llave primaria	
Descripción	Texto	[50]; Requerido	
SL	Entero	Requerido	Mínimo de distancia media entre caminos (metros)
SH	Entero	Requerido	Máximo de distancia media entre caminos (metros)
LL	Entero	Requerido	Mínimo de distancia media entre acopiaderos (metros)
LH	Entero	Requerido	Máximo de distancia media entre acopiaderos (metros)
VolMad	Simple	Requerido	Volumen de madera a aprovechar (m3/ha)

Objeto "***Secuencia***"

Atributo	Tipo	Restricción	Descripción
ID	Autonumérico	Llave primaria	
Tipo	Byte	Entre 0 y 1; Requerido	Determina si la secuencia es de Construcción de caminos y acopiaderos o de Aprovechamiento
Tecnología	Entero	Llave extranjera, relación 1 a n con Tecnología	

Objeto "***Equipos***"

Atributo	Tipo	Restricción	Descripción
ID	Autonumérico	Llave primaria	
Descripción	Texto	[25]; Requerido	
Tipo	Byte	Entre 0 y 3	Determina el tipo de equipo según especializaciones

Objeto "***Tractor***"

Atributo	**Tipo**	**Restricción**	**Descripción**
ID	Entero	Llave primaria	
FijPrecComp	Moneda	>= 0	Precio de compra de la máquina ($)
FijValResid	Moneda	>= 0	Valor residual ($)
FijVidaUtil	Simple	>= 0	Vida útil en años
FijDiaTrabAn	Simple	>= 0, <=365	Días de trabajo en el año
FijHorTrabDia	Simple	>= 0, <= 24	Horas de trabajo por día
FijTasaInter	Simple	>= 0, <=100	Tasa de interés (%)
FijTasaImpAn	Simple	>= 0, <=100	Tasa media anual para impuestos, licencia, seguro y protección (%)
OperPcDepHerrMantRep	Simple	>= 0, <=100	Depreciación de la herramienta por mantenimiento y reparación (%)
OperConsCombHor	Simple	>= 0	Consumo de combustible en litros por horas
OperValLitComb	Moneda	>= 0	Valor del litro de combustible ($)
OperPcConsCombEstConsLub	Simple	>=0, <=100	% de consumo de combustible para estimar consumo de lubricante
OperValLitLub	Moneda	>= 0	Valor del litro de lubricante ($)
OperValOtMat	Moneda	>= 0	Valor de otros materiales ($)
OperVidUtOtMatHor	Byte	>=0	Vida útil de otros materiales en horas
OperValCab	Moneda	>=0	Valor del cable ($)
OperVidUtCabHor	Simple	>=0	Vida útil del cable en horas
OperValWin	Moneda	>=0	Valor del winche ($)
OperVidUtWinHor	Simple	>=0	Vida útil del winche en horas
OperValOrugNeum	Moneda	>=0	Valor de orugas o neumáticos ($)
OperVidUtOrugNeumHor	Simple	>=0	Vida útil de orugas o neumáticos en horas
LabSalBasOper	Moneda	>=0	Salario básico del operador por horas ($)
LabSalBasAyud	Moneda	>=0	Salario básico del ayudante por horas ($)
LabCantAyud	Byte	>=0	Cantidad de ayudantes
LabBenAdic	Moneda	>=0	Beneficios adicionales ($)
LabHorNoTrab	Simple	>=0, <=24	Horas no trabajadas en la jornada
LabHorTrab	Simple	>=0, <=24	Horas de trabajo en la jornada
LabPCientCosDir	Simple	>=0, <=100	% de los costos directos de labor para supervisión
AlqCostFijo	Moneda	>=0	Costo fijo ($)
AlqCostOper	Moneda	>=0	Costo de operación ($)
AlqCostLabor	Moneda	>=0	Costo de labor ($)

Objeto "**Herramienta**"

Atributo	**Tipo**	**Restricción**	**Descripción**
ID	Entero		
FijPrecComp	Moneda	>= 0	Precio de compra de la máquina ($)
FijValResid	Moneda	>= 0	Valor residual ($)
FijVidaUtil	Simple	>= 0	Vida útil en años
FijDiaTrabAn	Simple	>= 0, <=365	Días de trabajo en el año
FijHorTrabDia	Simple	>= 0, <= 24	Horas de trabajo por día
FijTasaInter	Simple	>= 0, <=100	Tasa de interés (%)
FijTasaImpAn	Simple	>= 0, <=100	Tasa media anual para impuestos, licencia, seguro y protección (%)
OperPcDepHerrMantRep	Simple	>= 0, <=100	Depreciación de la herramienta por mantenimiento y reparación (%)
OperConsCombHor	Simple	>= 0	Consumo de combustible en litros por horas
OPERVALLITCOMB	Moneda	>= 0	Valor del litro de combustible ($)
OperPcConsCombEstConsLub	Byte	>= 0, <=100	% de consumo de combustible para estimar consumo de lubricante
OperValLitLub	Moneda	>= 0	Valor del litro de lubricante ($)
OperValOtMat	Moneda	>= 0	Valor de otros materiales ($)
OperVidUtOtMatHor	Simple	>= 0	Vida útil de otros materiales en horas
OperValSab	Moneda	>= 0	Valor del sable ($)
OperVidUtSabHor	Simple	>= 0	Vida útil del sable en horas
OperValCad	Moneda	>= 0	Valor de la cadena ($)
OperVidUtCadHor	Simple	>= 0	Vida útil de la cadena en horas
LabSalBasOper	Moneda	>= 0	Salario básico del operador por horas ($)
LabSalBasAyud	Moneda	>= 0	Salario básico del ayudante por horas ($)
LabCantAyud	Byte	>= 0	Cantidad de ayudantes
LabBenAdic	Moneda	>= 0	Beneficios adicionales ($)
LabHorNoTrab	Simple	>= 0, <= 24	Horas no trabajadas en la jornada
LabHorTrab	Simple	>= 0, <= 24	Horas de trabajo en la jornada
LabPCientCosDir	Byte	>= 0, <=100	% de los costos directos de labor para supervisión (%)
AlqCostFijo	Moneda	>= 0	Costo fijo ($)
AlqCostOper	Moneda	>= 0	Costo de operación ($)
AlqCostLabor	Moneda	>= 0	Costo de labor ($)

Objeto "***Camion***"

Atributo	**Tipo**	**Restricción**	**Descripción**
ID	Entero	Llave primaria	
FijPrecComp	Moneda	>= 0	Precio de compra de la máquina ($)
FijValResid	Moneda	>= 0	Valor residual ($)
FijVidaUtil	Simple	>= 0	Vida útil en años
FijDiaTrabAn	Simple	>= 0, <=365	Días de trabajo en el año
FijHorTrabDia	Simple	>= 0, <= 24	Horas de trabajo por día
FijTasaInter	Simple	>= 0, <=100	Tasa de interés (%)
FijTasaImpAn	Simple	>= 0, <=100	Tasa media anual para impuestos, licencia, seguro y protección (%)
OperPcDepHerrMantRep	Simple	>= 0, <=100	Depreciación de la herramienta por mantenimiento y reparación (%)
OperConsCombHor	Simple	>= 0	Consumo de combustible en litros por horas
OperValLitComb	Moneda	>= 0	Valor del litro de combustible ($)
OperPcConsCombEstConsLub	Simple	>=0, <=100	% de consumo de combustible para estimar consumo de lubricante
OperValLitLub	Moneda	>= 0	Valor del litro de lubricante ($)
OperValOtMat	Moneda	>= 0	Valor de otros materiales ($)
OperVidUtOtMatHor	Byte	>=0	Vida útil de otros materiales en horas
OperValNeum	Moneda	>=0	Valor de los neumáticos ($)
OperCantNeum	Byte	>=0	Cantidad de neumáticos
OperVidUtNeum	Simple	>=0	Vida útil de los neumáticos en horas
LabSalBasOper	Moneda	>=0	Salario básico del operador por horas ($)
LabSalBasAyud	Moneda	>=0	Salario básico del ayudante por horas ($)
LabCantAyud	Byte	>=0	Cantidad de ayudantes
LabBenAdic	Moneda	>=0	Beneficios adicionales ($)
LabHorNoTrab	Byte	>=0, <=24	Horas no trabajadas en la jornada
LabHorTrab	Simple	>=0, <=24	Horas de trabajo en la jornada
LabPCientCosDir	Simple	>=0, <=100	% de los costos directos de labor para supervisión (%)
AlqCostFijo	Moneda	>=0	Costo fijo ($)
AlqCostOper	Moneda	>=0	Costo de operación ($)
AlqCostLabor	Moneda	>=0	Costo de labor ($)

Objeto "***Animal***"

Atributo	**Tipo**	**Restricción**	**Descripción**
ID	Entero	Llave primaria	
FijPrecComp	Moneda	>= 0	Precio de compra de la máquina ($)
FijValResid	Moneda	>= 0	Valor residual ($)
FijVidaUtil	Simple	>= 0	Vida útil en años
FijDiaTrabAn	Simple	>= 0, <=365	Días de trabajo en el año
FijHorTrabDia	Simple	>= 0, <= 24	Horas de trabajo por día
FijTasaInter	Simple	>= 0, <=100	Tasa de interés (%)
FijTasaImpAn	Simple	>= 0, <=100	Tasa media anual para impuestos, licencia, seguro y protección (%)
FijValAlimAnim	Moneda	>= 0	Valor de alimentación normal de los animales ($)
FijValMedVeter	Moneda	>= 0	Valor de medicamentos y servicios veterinarios ($)
OperValCompArn	Moneda	>= 0	Valor de compra del arnés ($)
OperValResArn	Moneda	>= 0	Valor residual del arnés ($)
OperVidUtArn	Simple	>= 0	Vida útil del arnés
OperValCompMis	Moneda	>= 0	Valor de compra de misceláneas ($)
OperValResMis	Moneda	>= 0	Valor residual de misceláneas ($)
OperVidUtMisAn	Simple	>= 0	Vida útil de misceláneas en años
OperValAlimSupAn	Moneda	>= 0	Valor de la alimentación suplementaria por año ($)
OperValMedSupAn	Moneda	>= 0	Valor de medicamentos suplementarios por años ($)
LabSalBasOper	Moneda	>= 0	Salario básico del operador por horas ($)
LabSalBasAyud	Moneda	>= 0	Salario básico del ayudante por horas ($)
LabCantAyud	Byte	>= 0	Cantidad de ayudantes
LabBenAdic	Moneda	>= 0	Beneficios adicionales ($)
LabHorNoTrab	Simple	>= 0, <= 24	Horas no trabajadas en la jornada
LabHorTrab	Byte	>= 0, <= 24	Horas de trabajo en la jornada
LabPCientCosDir	Byte	>= 0, <= 100	% de los costos directos de labor para supervisión (%)
AlqCostFijo	Moneda	>=0	Costo fijo ($)
AlqCostOper	Moneda	>=0	Costo de operación ($)
AlqCostLabor	Moneda	>=0	Costo de labor ($)

Objeto "***Etapa***"

Atributo	**Tipo**	**Restricción**	**Descripción**
ID	Entero, Autonumérico	Llave primaria	

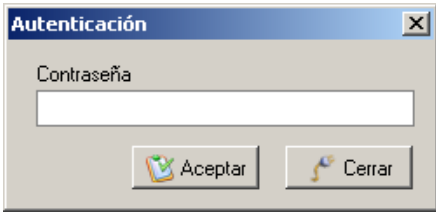
Operación	Byte	>= 0	Tipo de operación realizada
Secuencia	Entero largo	Llave extranjera, relación 1 a n con *Secuencia*	
Equipo	Entero largo	Llave extranjera, relación 1 a n con *Equipo*	
FormulaDeRendimiento	Texto	[255], Requerido	Expresión para calcular rendimiento de la etapa
Alquilado	Lógico		
X	Entero largo	Requerido	Coordenadas en eje x del componente que representa una etapa dentro del formulario contenedor.
Y	Entero largo	Requerido	Coordenadas en eje y del componente que representa una etapa dentro del formulario contenedor.

Objeto "***ParametroDeRendimiento***"

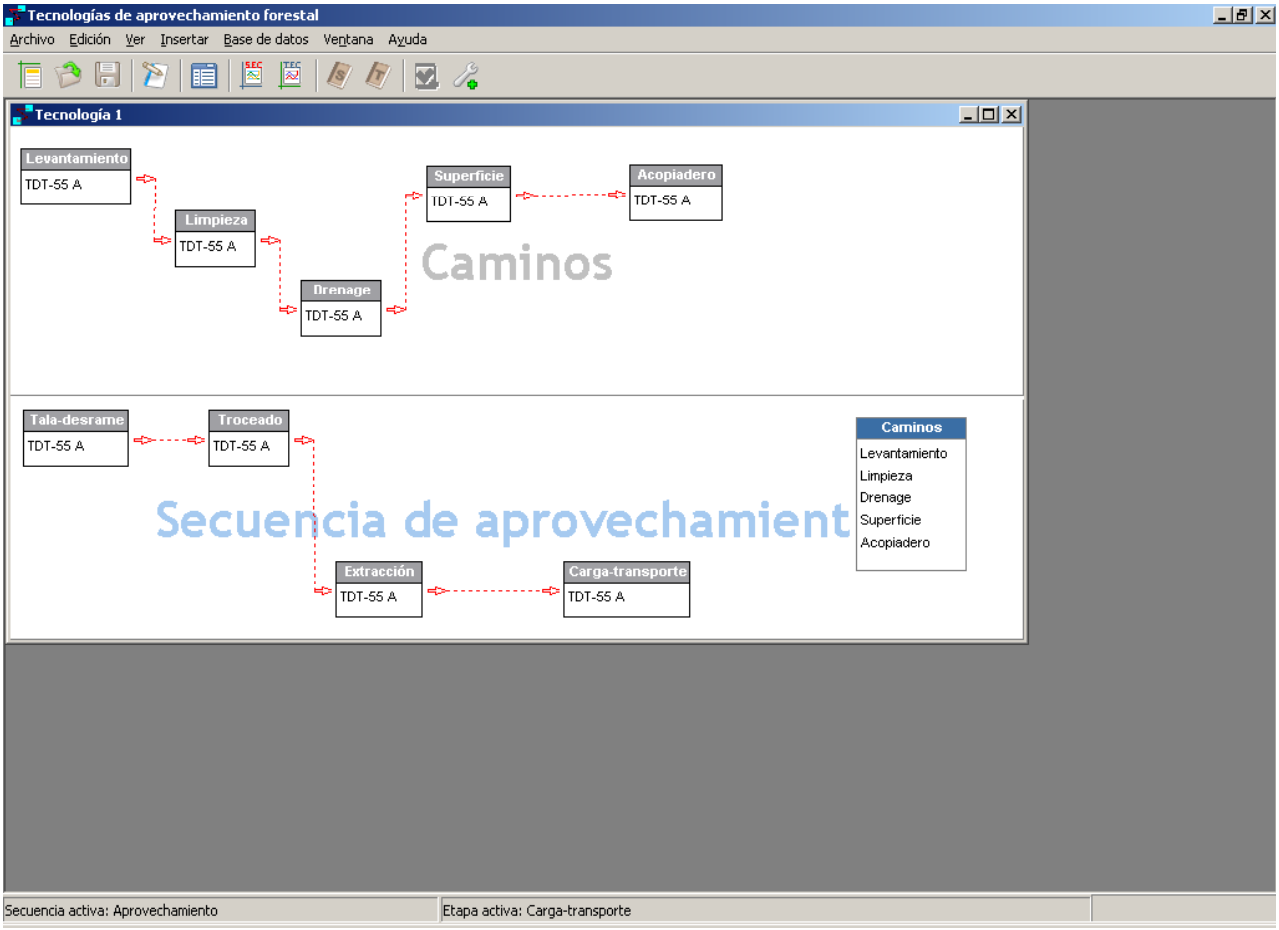
Atributo	**Tipo**	**Restricción**	**Descripción**
ID	Entero, Autonumérico	Llave primaria	
Nombre	Texto	[25], Requerido	Nombre del parámetro
Valor	Simple		Valor del parámetro
Descripción	Texto	[50]	
Etapa	Entero largo	Llave extranjera, relación 1 a n con *Etapa*	

APÉNDICE B.

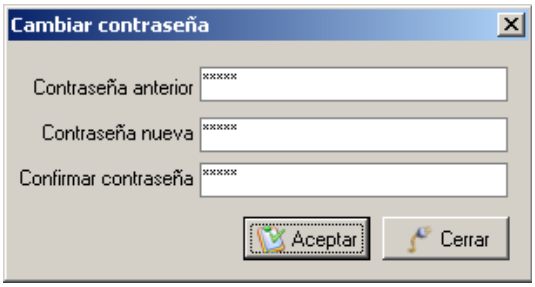
Interfases del sistema APROV PLUS



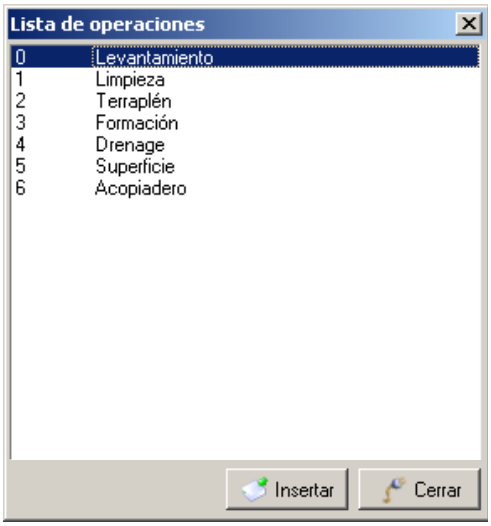
Interfase A. Autentificación de APROV PLUS



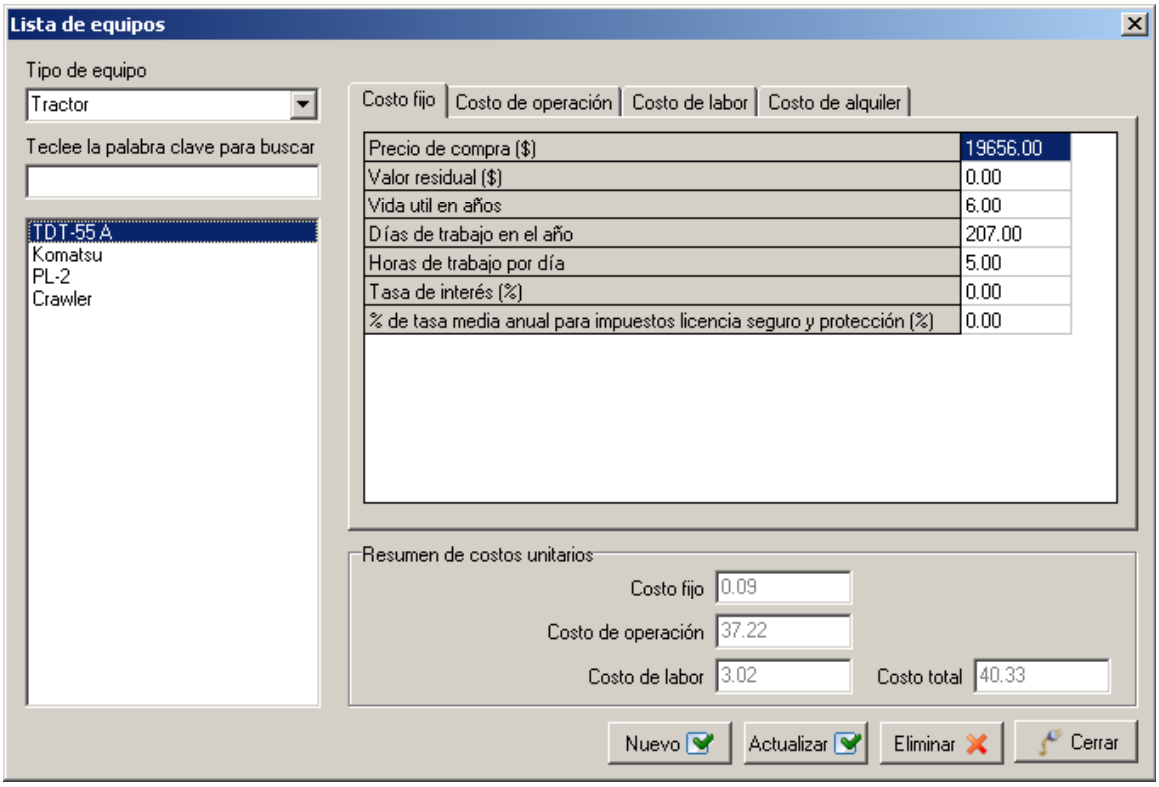
Interfase IP. Ventana Principal de APROV PLUS



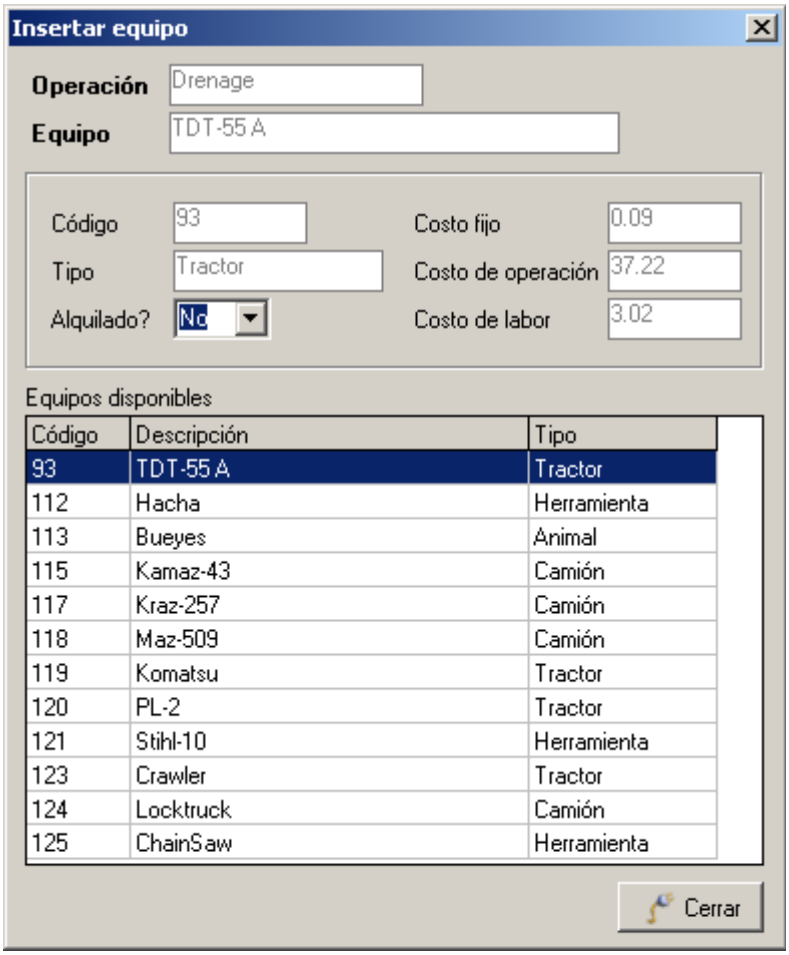
Interfase C. Cambiar contraseña de APROV PLUS.



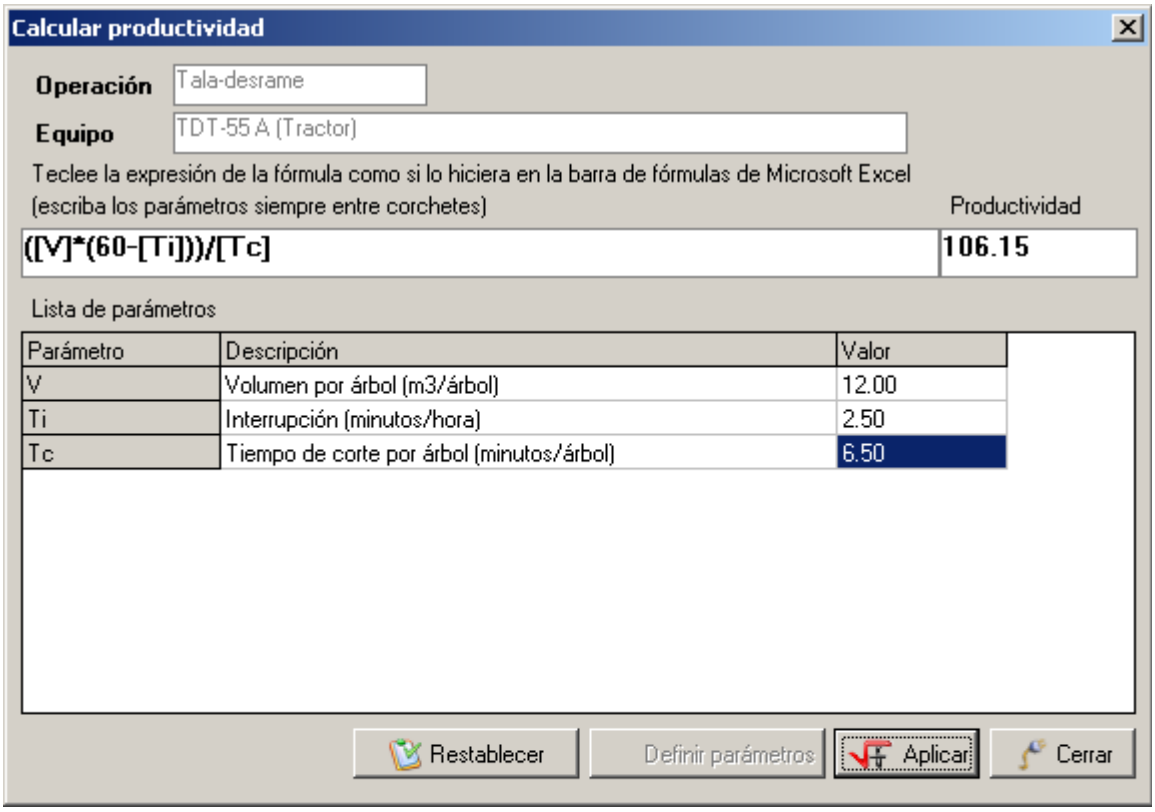
Interfase L. Incluir operaciones de APROV PLUS



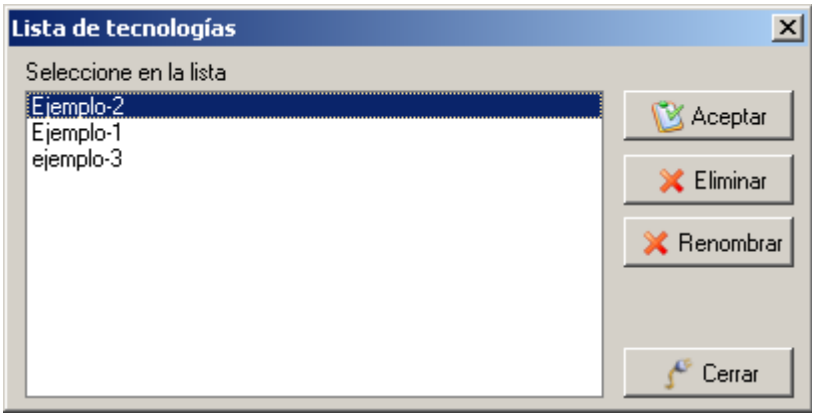
Interfase E. Actualizar Registro de Equipos de APROV PLUS



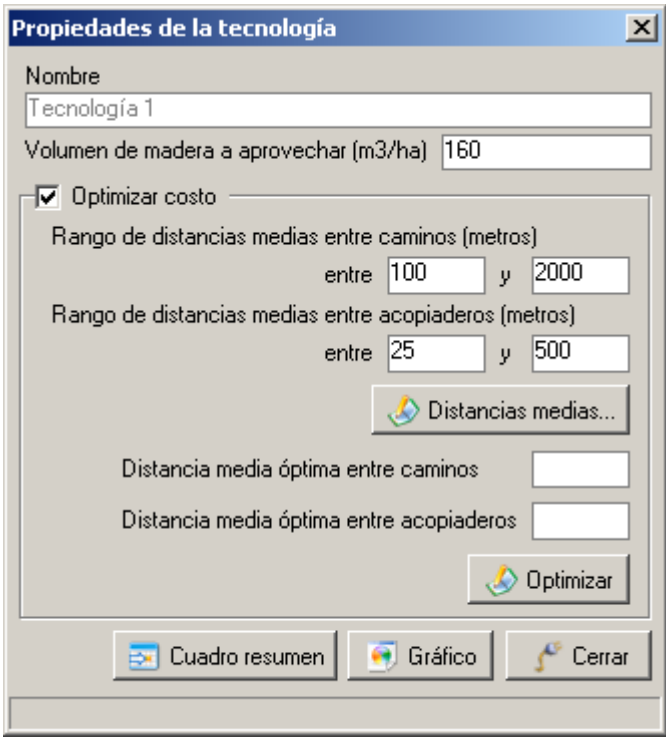
Interfase AE. Asignar equipos de APROV PLUS



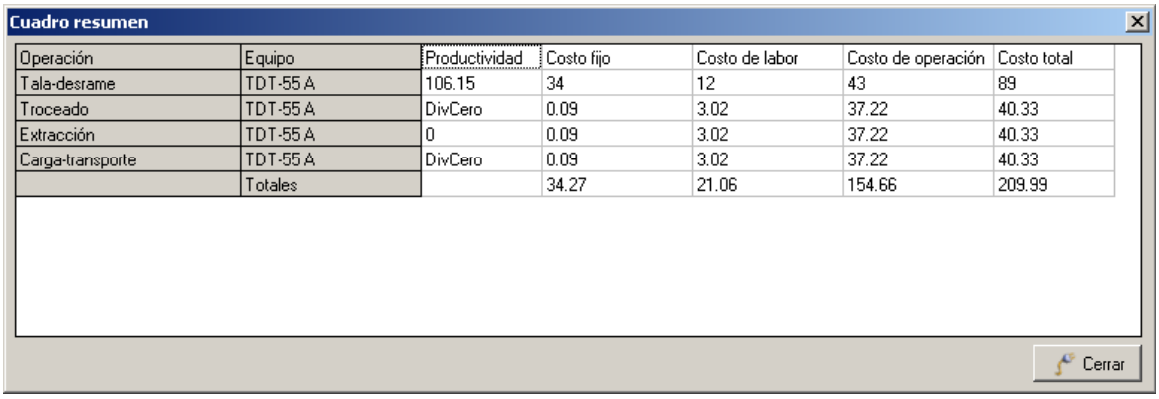
Interfase P. Asignar valores a parámetros de fórmula de rendimiento de APROV PLUS



Interfase T. Cargar tecnología de aprovechamiento de APROV PLUS

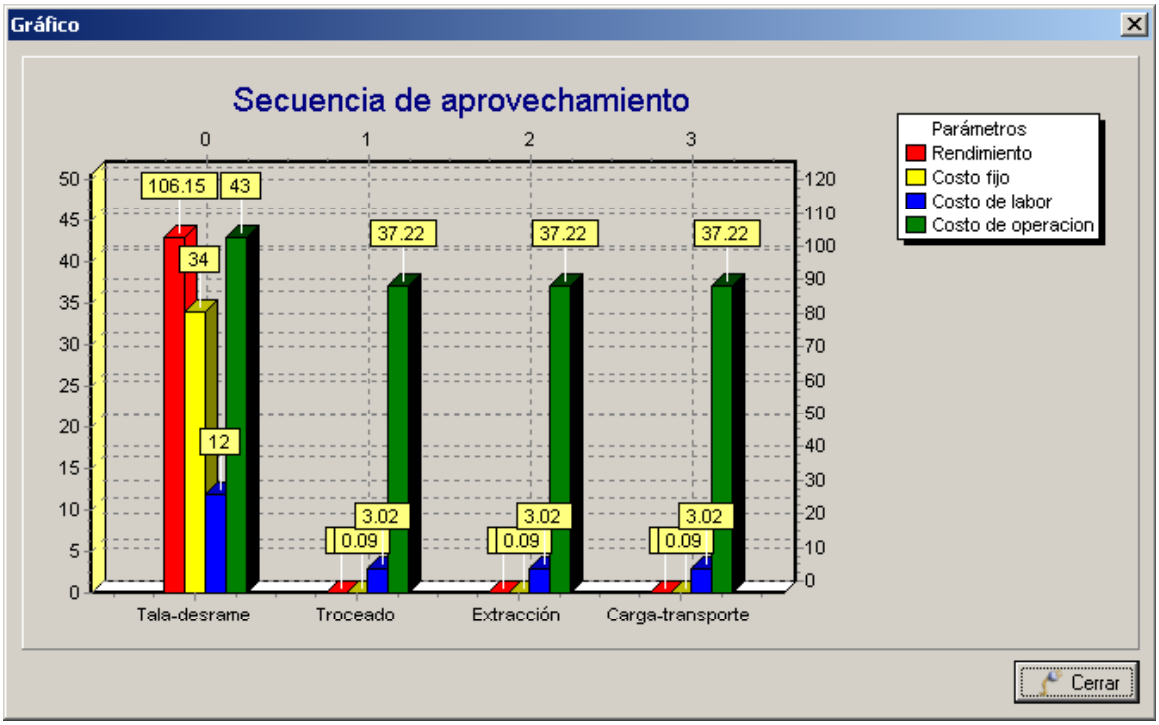


Interfase O. Propiedades de la tecnología de aprovechamiento de APROV PLUS



Operación	Equipo	Productividad	Costo fijo	Costo de labor	Costo de operación	Costo total
Tala-desrame	TDT-55 A	106.15	34	12	43	89
Troceado	TDT-55 A	DivCero	0.09	3.02	37.22	40.33
Extracción	TDT-55 A	0	0.09	3.02	37.22	40.33
Carga-transporte	TDT-55 A	DivCero	0.09	3.02	37.22	40.33
	Totales		34.27	21.06	154.66	209.99

Interfase R. Cuadro resumen de APROV PLUS



Interfase G. Gráfico de APROV PLUS